



**COORDINATED HIGHWAYS ACTION RESPONSE TEAM
STATE HIGHWAY ADMINISTRATION**

CHART R3B3 Detailed Design

**Contract SHA-06-CHART
Document # WO15-DS-001
Work Order 15, Deliverable 04**

December 23, 2008

**By
CSC**



Table of Contents

1	Introduction.....	1-1
1.1	Purpose	1-1
1.2	Objectives	1-3
1.3	Scope	1-3
1.4	Design Process.....	1-3
1.5	Design Tools	1-3
1.6	Work Products.....	1-3
2	Architecture	2-1
2.1	Network/Hardware.....	2-1
2.2	Software.....	2-1
2.2.1	COTS Products	2-1
2.2.2	Deployment /Interface Compatibility	2-3
2.3	Security.....	2-9
2.4	Data.....	2-11
2.4.1	Data Storage.....	2-11
2.4.2	Database Design	2-35
3	Key Design Concepts.....	3-1
3.1	Travel Routes	3-1
3.2	Traveler Information Message Templates.....	3-1
3.3	Traveler Information Messages.....	3-2
3.4	External Interface to INRIX.....	3-2
3.5	External Interface to Vector	3-3
3.6	External Interface to RITIS.....	3-3
3.7	CHART Data Export.....	3-4
3.8	External System Connection Status	3-5
3.9	Device Locations	3-5
3.10	TCP/IP DMS and TSS Communications	3-5
3.11	Error Processing	3-6
3.12	Packaging	3-6
3.13	Assumptions and Constraints	3-9
4	Use Cases	4-1
4.1	High Level	4-1
4.1.1	R3B3HighLevel (Use Case Diagram).....	4-1

4.2	Travel/Toll Routes	4-5
4.2.1	R3B3ManageTravelRoutes (Use Case Diagram)	4-5
4.2.2	R3B3ImportInrixData (Use Case Diagram)	4-10
4.2.3	R3B3ImportVectorData (Use Case Diagram)	4-13
4.2.4	R3B3ManageTravelerInformationMessages (Use Case Diagram).....	4-17
4.2.5	R3B3ManageDeviceQueue (Use Case Diagram)	4-23
4.2.6	R3B3ConfigureDMSTravelerInfoMsgSettings (Use Case Diagram).....	4-28
4.3	Device Enhancements	4-31
4.3.1	R3B3ConfigureDevices (Use Case Diagram).....	4-31
4.3.2	R3B3ViewDeviceLists (Use Case Diagram).....	4-34
4.3.3	R3B3ViewDeviceDetails (Use Case Diagram)	4-37
4.3.4	R3B3ManageTrafficEvents (Use Case Diagram).....	4-40
4.3.5	R3B3ManageDevices (Use Case Diagram).....	4-44
4.4	External Interfaces (RITIS Import).....	4-46
4.4.1	R3B3ImportRITISData (Use Case Diagram)	4-46
4.4.2	R3B3ConfigureExternalSystemSettings (Use Case Diagram)	4-50
4.5	Public/Private Data Sharing	4-55
4.5.1	R3B3ManageUsers (Use Case Diagram).....	4-55
4.5.2	R3B3ProvideDataToExternalSystems (Use Case Diagram).....	4-59
4.6	Alerts/Notifications	4-63
4.6.1	R3B3ManageAlertsAndNotifications (Use Case Diagram)	4-63
4.7	Configure system	4-68
4.7.1	R3B3ConfigureSystem (Use Case Diagram).....	4-68
5	Detailed Design.....	5-1
5.1	Human-Machine Interface.....	5-1
5.1.1	Travel Routes.....	5-1
5.1.2	DMS Message Templates	5-16
5.1.3	Travel Time and Toll Rate Messages	5-24
5.1.4	DMS Travel Time / Toll Rate Settings	5-29
5.1.5	System Profile Travel Time Settings	5-32
5.1.6	Geographical Settings	5-36
5.1.7	External Events.....	5-41
5.1.8	External Devices	5-45
5.1.9	External System Related Settings	5-51
5.1.10	External System Connection Status	5-58
5.1.11	Alerts	5-59

5.1.12	Device Locations	5-65
5.1.13	Miscellaneous Device Enhancements	5-69
5.1.14	Public / Private Data Sharing	5-72
5.2	Alert Module	5-74
5.2.1	Classes	5-74
5.3	Camera Control Module	5-88
5.3.1	Classes	5-88
5.3.2	Sequence Diagrams	5-112
5.4	DMS Control Module	5-114
5.4.1	Classes	5-114
5.4.2	Sequence diagrams	5-132
5.5	DMS Protocols Pkg	5-165
5.5.1	Classes	5-165
5.5.2	Sequence Diagrams	5-165
5.6	DMSUtilityPkg	5-171
5.6.1	Classes	5-171
5.6.2	Sequence diagrams	5-179
5.7	DeviceUtilityPkg	5-182
5.7.1	Classes	5-182
5.7.2	Sequence Diagrams	5-185
5.8	External Interface Module	5-191
5.8.1	Classes	5-191
5.8.2	Sequence Diagrams	5-221
5.9	GeoAreaModulePkg	5-255
5.9.1	Classes	5-255
5.9.2	Sequence Diagrams	5-258
5.10	HARControlModulePkg	5-264
5.10.1	Classes	5-264
5.10.2	Sequence Diagrams	5-274
5.11	INRIXDataImportModule	5-277
5.11.1	Classes	5-277
5.11.2	SequenceDiagram	5-282
5.12	INRIXLinkImportProgramPkg	5-288
5.12.1	Classes	5-288
5.12.2	Sequence Diagrams	5-290
5.13	Java Classes	5-292
5.13.1	Classes	5-292

5.14 MessageTemplateModulePkg	5-297
5.14.1 Classes	5-297
5.14.2 Sequence Diagrams.....	5-300
5.15 RoadwayLocationModule	5-308
5.15.1 Classes	5-308
5.15.2 Sequence Diagrams.....	5-311
5.16 SHAZAMControlModulePkg.....	5-314
5.16.1 Classes	5-314
5.16.2 SequenceDiagrams.....	5-322
5.17 SHAZAMManagementPkg.....	5-323
5.17.1 Classes	5-323
5.18 SystemInterfaces	5-324
5.18.1 Classes	5-324
5.19 TSSManagementModulePkg	5-418
5.19.1 Classes	5-418
5.19.2 SequenceDiagrams.....	5-431
5.20 TravelRouteModulePkg	5-437
5.20.1 Classes	5-437
5.20.2 SequenceDiagrams.....	5-447
5.21 UserManagementmodulePkg.....	5-469
5.21.1 Classes	5-469
5.22 UtilityPkg.....	5-471
5.22.1 Classes	5-471
5.22.2 SequenceDiagrams.....	5-485
5.23 UtilityPkg.wrappers.....	5-486
5.23.1 Classes	5-486
5.24 Webservices.WSTrafficEventExportModule	5-490
5.24.1 Classes	5-490
5.24.2 SequenceDiagrams.....	5-494
5.25 WSDMSEExportModulePkg.....	5-499
5.25.1 Classes	5-499
5.25.2 Sequence Diagrams.....	5-504
5.26 WebServices.TollrateImportModule	5-511
5.26.1 Classes	5-511
5.26.2 Sequence Diagrams.....	5-517
5.27 Webservices.base	5-522
5.27.1 Classes	5-522

5.27.2	Sequence Diagrams.....	5-527
5.28	Chartlite.data	5-531
5.28.1	Classes	5-531
5.29	Chartlite.data.dms-data	5-538
5.29.1	Classes	5-538
5.29.2	Sequence Diagrams.....	5-542
5.30	Chartlite.data.video-data	5-549
5.30.1	Classes	5-549
5.31	Chartlite.data.location-data.....	5-552
5.31.1	Classes	5-552
5.32	Chartlite.data.shazam-data.....	5-553
5.32.1	Classes	5-553
5.33	Chartlite.data.har-data	5-555
5.33.1	Classes	5-555
5.34	Chartlite.data.arbqueue-data	5-559
5.34.1	Classes	5-559
5.35	Chartlite.data.travelroutes-data.....	5-560
5.35.1	Classes	5-560
5.35.2	Sequence diagrams	5-568
5.36	Chartlite.data.templates-data	5-584
5.36.1	Classes	5-584
5.36.2	Sequence Diagrams.....	5-588
5.37	Chartlite.data.geoareas-data.....	5-591
5.37.1	Classes	5-591
5.37.2	Sequence Diagrams.....	5-593
5.38	Chartlite.data.alerts-data.....	5-594
5.38.1	Classes	5-594
5.39	Chartlite.data.externalsystem-data	5-598
5.39.1	Classess.....	5-598
5.39.2	Sequence Diagrams.....	5-600
5.40	Chartlite.data.tss-data	5-604
5.40.1	Classes	5-604
5.41	Chartlite.data.trafficevents-data	5-606
5.41.1	Classes	5-606
5.42	Chartlite.data.plans-data	5-612
5.42.1	Classes	5-612
5.43	Chartlite.servlet	5-615

5.43.1	Classes	5-615
5.43.2	Sequence Diagrams.....	5-619
5.44	Chartlite.servlet.usermgmt	5-620
5.44.1	Class Diagrams	5-620
5.44.2	Sequence diagrams	5-621
5.45	Chartlite.servlet.tss.....	5-638
5.45.1	Classes	5-638
5.45.2	Sequence Diagrams.....	5-642
5.46	Chartlite.servlet.servlet-dynlist	5-648
5.46.1	Classes	5-648
5.46.2	Sequence Diagrams.....	5-652
5.47	Chartlite.servlet.dms	5-654
5.47.1	Class diagrams	5-654
5.47.2	Sequence Diagrams.....	5-658
5.48	Chartlite.servlet.alerts.....	5-683
5.48.1	Sequence Diagrams.....	5-683
5.49	Chartlite.servlet.geoareamgmt	5-685
5.49.1	Class Diagrams	5-685
5.49.2	Sequence Diagrams.....	5-686
5.50	Chartlite.servlet.travelroutes.....	5-690
5.50.1	Class Diagrams	5-690
5.50.2	Sequence Diagrams.....	5-695
5.51	Chartlite.servlet.externalsystemmgmt	5-722
5.51.1	ClassFiles	5-722
5.51.2	Sequence Diagrams.....	5-725
5.52	chartlite.servlet.messagesemplates	5-741
5.52.1	Class Diagrams	5-741
5.52.2	Sequence Diagrams.....	5-745
5.53	Chartlite.servlet.video	5-749
5.53.1	Class Diagrams	5-749
5.53.2	Sequence Diagrams.....	5-751
5.54	Chartlite.servlet.trafficevents	5-755
5.54.1	Class Diagrams	5-755
5.54.2	Sequence Diagrams.....	5-760
5.55	Chartlite.servlet.trafficevents.location.....	5-781
5.55.1	Classes	5-781
5.55.2	Sequence Diagrams.....	5-783

5.56	Chartlite.servlet.shazam.....	5-786
5.56.1	Class Diagrams	5-786
5.56.2	Sequence Diagrams.....	5-788
5.57	Chartlite.servlet.har	5-795
5.57.1	Class diagrams	5-795
5.57.2	Sequence Diagrams.....	5-798
5.58	Chartlite.Flex.shared.components-flex.....	5-802
5.58.1	Class Diagrams	5-802
5.59	Chartlite.Flex.editlocation.....	5-803
5.59.1	Class Diagrams	5-803
5.60	Chartlite.util	5-805
5.60.1	Classes	5-805
5.60.2	Sequence Diagrams.....	5-807
5.61	Chartlite.util.dynlist	5-808
5.61.1	Class Diagrams	5-808
6	Mapping To Requirements.....	6-1
7	Acronyms/Glossary	7-1

Table of Figures

Figure 2-1 CHART and External Interfaces	2-5
Figure 2-2 CHART R3B3 External Interface Deployment	2-6
Figure 2-3 CHART Internal Interfaces (GUI Deployment).....	2-8
Figure 2-4 CHART Internal Interfaces (Server Deployment)	2-9
Figure 2-5 CHART R3B3 Database Architecture	2-12
Figure 4-1 R3B3ManageTravelRoutes (Use Case Diagram)	4-5
Figure 4-2 R3B3ImportInrixData (Use Case Diagram).....	4-10
Figure 4-3 R3B3ImportVectorData (Use Case Diagram).....	4-13
Figure 4-4 R3B3ManageTravelerInformationMessages (Use Case Diagram)	4-17
Figure 4-5 R3B3ManageDeviceQueue (Use Case Diagram)	4-23
Figure 4-6 R3B3ConfigureDMSTravelerInfoMsgSettings (Use Case Diagram).....	4-28
Figure 4-7 R3B3ConfigureDevices (Use Case Diagram).....	4-31
Figure 4-8 R3B3ViewDeviceLists (Use Case Diagram)	4-34
Figure 4-9 R3B3ViewDeviceDetails (Use Case Diagram).....	4-37
Figure 4-10 R3B3ManageTrafficEvents (Use Case Diagram)	4-40
Figure 4-11 R3B3ManageDevices (Use Case Diagram)	4-44
Figure 4-12 R3B3ImportRITISData (Use Case Diagram)	4-46
Figure 4-13 R3B3ConfigureExternalSystemSettings (Use Case Diagram).....	4-50
Figure 4-14 R3B3ManageUsers (Use Case Diagram).....	4-55
Figure 4-15 R3B3ProvideDataToExternalSystems (Use Case Diagram).....	4-59
Figure 4-16 R3B3ManageAlertsAndNotifications (Use Case Diagram).....	4-63
Figure 4-17 R3B3ConfigureSystem (Use Case Diagram)	4-68
Figure 5-1 View Travel Routes menu item.....	5-1
Figure 5-2 View Travel Routes page.....	5-2
Figure 5-3 Set Travel Route List Columns	5-3
Figure 5-4 Add Travel Route.....	5-5
Figure 5-5 Travel Route Links - Empty List	5-7
Figure 5-6 Select Link, No Prior Link Selected	5-7
Figure 5-7 Select Link – Search Results.....	5-8
Figure 5-8 Travel Route Links - Populated List	5-8
Figure 5-9 Link Settings for Travel Route.....	5-9
Figure 5-10 Select Link - Prior Link Exists.....	5-10
Figure 5-11 Select Link - Search for Links	5-10
Figure 5-12 Toll Rate Source Unspecified	5-11
Figure 5-13 Select Toll Rate Source.....	5-11
Figure 5-14 Toll Rate Source Specified	5-11
Figure 5-15 Travel Route Details - Status Section	5-12
Figure 5-16 Travel Route Details - Link Status Section.....	5-12
Figure 5-17 Travel Route Details - Link History Section.....	5-13
Figure 5-18 Travel Route Details - Toll Rate History Section	5-13
Figure 5-19 Travel Route Details - Link Configuration Section	5-13
Figure 5-20 Travel Route Details - Toll Rate Configuration Section.....	5-14
Figure 5-21 Travel Route Details - General Settings Section.....	5-14
Figure 5-22 Travel Route Details - Location Settings Section	5-14
Figure 5-23 Travel Route Details - Travel Time Settings Section.....	5-14
Figure 5-24 Travel Route Details - Toll Rate Settings Section	5-15
Figure 5-25 Link Details.....	5-15
Figure 5-26 View / Edit DMS Message Templates Link.....	5-16
Figure 5-27 Travel Time / Toll Message Template List.....	5-16
Figure 5-28 Template List Column Settings.....	5-18
Figure 5-29 Add Message Template - Select Sign Size.....	5-19
Figure 5-30 Add DMS Message Template	5-20
Figure 5-31 DMS Message Templates - Adding Data Fields	5-21

Figure 5-32 DMS Message Templates - Default Editor	5-23
Figure 5-33 Travel Time / Toll Rate Message List on DMS Details Page	5-24
Figure 5-34 Add Travel Time / Toll Rate Message - Initial Form.....	5-25
Figure 5-35 Add Travel Time / Toll Rate Message - Template Selected	5-26
Figure 5-36 DMS Arbitration Queue With Toll Rate Message	5-28
Figure 5-37 DMS Travel Time / Toll Rate Arb Queue Levels	5-30
Figure 5-38 DMS Travel Time Message Schedule.....	5-30
Figure 5-39 DMS Travel Time Message Schedule Form	5-31
Figure 5-40 DMS Associated Travel Routes - DMS Details Page	5-32
Figure 5-41 DMS Associated Travel Routes - Edit Form	5-32
Figure 5-42 Travel Time System Profile Settings	5-33
Figure 5-43 Configure Travel Time Range Settings.....	5-34
Figure 5-44 System Wide Travel Time Message Schedule	5-35
Figure 5-45 Miscellaneous Travel Time Settings.....	5-36
Figure 5-46 Geographical Settings	5-37
Figure 5-47 Geographical Areas – List.....	5-38
Figure 5-48 Add Geographical Area	5-39
Figure 5-49 Miscellaneous Geographical Settings	5-40
Figure 5-50 External Event Rules Link	5-41
Figure 5-51 Add External Event Inclusion Rule.....	5-43
Figure 5-52 External Device Management Links	5-46
Figure 5-53 Manage External DMSs - Query Page	5-47
Figure 5-54 Manage External DMSs - Search Results	5-48
Figure 5-55 Message Signs Link on Home Page.....	5-49
Figure 5-56 DMS List with External DMSs	5-49
Figure 5-57 Detectors Link on Home Page	5-50
Figure 5-58 TSS List with External TSSs	5-50
Figure 5-59 External System Alert and Notification Settings	5-51
Figure 5-60 External System Alert and Notification Settings - Edit	5-52
Figure 5-61 External Agency / Organization Mapping	5-53
Figure 5-62 External Client Management - view/edit Link.....	5-53
Figure 5-63 External Client List.....	5-54
Figure 5-64 Add External System Client.....	5-55
Figure 5-65 Set External Client Role(s)	5-56
Figure 5-66 Edit External System Client.....	5-57
Figure 5-67 External System Connection Status – Link.....	5-58
Figure 5-68 External System Connection Status	5-58
Figure 5-69 R3B3 Alerts on the Home Page	5-59
Figure 5-70 External Connection Alert Details - Type Specific Fields	5-60
Figure 5-71 External Event Alert Details - Type Specific Fields	5-61
Figure 5-72 Toll Rate Alert Details - Type Specific Fields.....	5-61
Figure 5-73 Travel Time Alert Details - Type Specific Fields	5-61
Figure 5-74 Configure Alert Audio Cues	5-63
Figure 5-75 Alert Timeout and Policy Settings	5-64
Figure 5-76 Location Settings Form.....	5-66
Figure 5-77 Device List with Location Columns	5-67
Figure 5-78 Device List - Set Column Visibility.....	5-67
Figure 5-79 Location Fields on Device Details Page	5-68
Figure 5-80 Devices Close to Traffic Event	5-69
Figure 5-81 Field Comm Settings - TCP/IP	5-70
Figure 5-82 NTCIP DMS Font and Line Spacing Settings	5-71
Figure 5-83 Lane Level Detector Data	5-72
Figure 5-84 Incident Name with Fatality.....	5-72
Figure 5-85 Incident Details with Fatality	5-73
Figure 5-86 Event History Link.....	5-73
Figure 5-87 Summary and Detailed Speed Data.....	5-73

Figure 5-88 Traffic Parameters showing Detailed Data	5-74
Figure 5-89 Sensitive Device Configuration Data	5-74
Figure 5-90 AlertModule (Class Diagram)	5-76
Figure 5-91 ProxyAlertClasses (Class Diagram)	5-84
Figure 5-92 VideoHighLevel (Class Diagram)	5-89
Figure 5-93 VideoHighLevel-VideoSource (Class Diagram)	5-97
Figure 5-94 CameraControlModule (Class Diagram)	5-103
Figure 5-95 CameraControlModule:SetCameraConfiguration (Sequence Diagram)	5-113
Figure 5-96 DMSControlClassDiagram (Class Diagram)	5-114
Figure 5-97 DMSControlClassDiagram-ExternalDMS (Class Diagram)	5-125
Figure 5-98 QueueableCommandClassDiagram (Class Diagram)	5-128
Figure 5-99 CHART2DMSImpl:validate (Sequence Diagram)	5-132
Figure 5-100 Chart2DMSFactoryImpl:checkTravInfoMsgSchedule (Sequence Diagram)	5-133
Figure 5-101 Chart2DMSImpl:RouteUpdate (Sequence Diagram)	5-134
Figure 5-102 Chart2DMSImpl:addDMSTravInfoMsg (Sequence Diagram)	5-135
Figure 5-103 Chart2DMSImpl:computeTravInfoMsgSchedEnabled (Sequence Diagram)	5-136
Figure 5-104 Chart2DMSImpl:modifyDMSTravInfoMsg (Sequence Diagram)	5-137
Figure 5-105 Chart2DMSImpl:removeDMSTravInfoMsg (Sequence Diagram)	5-138
Figure 5-106 Chart2DMSImpl:routeDeleted (Sequence Diagram)	5-139
Figure 5-107 Chart2DMSImpl:setQueueLevels (Sequence Diagram)	5-140
Figure 5-108 Chart2DMSImpl:setRelatedRoutes (Sequence Diagram)	5-141
Figure 5-109 Chart2DMSImpl:setTravInfoMsgEnabledFlag (Sequence Diagram)	5-142
Figure 5-110 Chart2DMSImpl:setTravelTimeSchedule (Sequence Diagram)	5-143
Figure 5-111 DMSControlModule:FmsGetConnectedPort (Sequence Diagram)	5-144
Figure 5-112 DMSControlModule:FmsReleasePort (Sequence Diagram)	5-145
Figure 5-113 DMSControlModule:HandleOpStatus (Sequence Diagram)	5-147
Figure 5-114 DMSControlModule:Initialize (Sequence Diagram)	5-149
Figure 5-115 DMSControlModule:RestoreDMS (Sequence Diagram)	5-151
Figure 5-116 DMSControlModule:SetConfiguration (Sequence Diagram)	5-153
Figure 5-117 DMSControlModule:Shutdown (Sequence Diagram)	5-154
Figure 5-118 DMSImpl:setLocation (Sequence Diagram)	5-155
Figure 5-119 DMSTravInfoMsgHandler:checkMessage (Sequence Diagram)	5-156
Figure 5-120 DMSTravInfoMsgDataSupplier:getData (Sequence Diagram)	5-157
Figure 5-121 EnabledTravInfoMsgCmd:execute (Sequence Diagram)	5-158
Figure 5-122 ExternalDMSImpl:GetExternalConfiguration (Sequence Diagram)	5-159
Figure 5-123 ExternalDMS:GetStatus (Sequence Diagram)	5-160
Figure 5-124 ExternalDMS:RemoveDMS (Sequence Diagram)	5-161
Figure 5-125 ExternalDMS:SetExternalConfiguration (Sequence Diagram)	5-162
Figure 5-126 ExternalDMS:updateStatus (Sequence Diagram)	5-163
Figure 5-127 chartlite.servlet.dms:setDMSConfigCommSettings (Sequence Diagram)	5-164
Figure 5-128 DMSProtocolsPkg:TypicalSetMessage (Sequence Diagram)	5-166
Figure 5-129 FP9500ProtocolHdlr:GetStatus (Sequence Diagram)	5-167
Figure 5-130 FP9500ProtocolHdlr:PixelTest (Sequence Diagram)	5-168
Figure 5-131 NTCIPProtocolHdlr:SetMessage (Sequence Diagram)	5-169
Figure 5-132 TS3001ProtocolHdlr:GetStatus (Sequence Diagram)	5-170
Figure 5-133 DMSUtility (Class Diagram)	5-172
Figure 5-134 DMSTravInfoMsgFormattingClasses (Class Diagram)	5-176
Figure 5-135 DMSTravInfoMsgTemplateModel:formatMulti (Sequence Diagram)	5-179
Figure 5-136 DMSTravInfoMsgTemplateModel:formatPageMulti (Sequence Diagram)	5-180
Figure 5-137 TemplateRow:formatMulti (Sequence Diagram)	5-181
Figure 5-138 . PortLocatorClasses (Class Diagram)	5-182
Figure 5-139 PortLocator:ReleasePort (Sequence Diagram)	5-185
Figure 5-140 PortLocator:ReleasePort2 (Sequence Diagram)	5-186
Figure 5-141 PortLocator:getConnectedPort (Sequence Diagram)	5-188
Figure 5-142 PortLocator:getConnectedPort2 (Sequence Diagram)	5-189
Figure 5-143 PortLocator:getPort (Sequence Diagram)	5-190

Figure 5-144 DMSImportAcquireClasses (Class Diagram)	5-191
Figure 5-145 DMSImportChartClasses (Class Diagram)	5-194
Figure 5-146 DMSImportModuleClasses (Class Diagram).....	5-197
Figure 5-147 DMSImportTranslationClasses (Class Diagram).....	5-201
Figure 5-148 EventAtisImportChartClasses (Class Diagram).....	5-203
Figure 5-149 ExternalDeviceManagerClasses (Class Diagram).....	5-206
Figure 5-150 ExternalSystemConnectionClasses (Class Diagram).....	5-208
Figure 5-151 TSSImportAcquireClasses (Class Diagram).....	5-210
Figure 5-152 TSSImportChartClasses (Class Diagram).....	5-213
Figure 5-153 TSSImportModuleClasses (Class Diagram)	5-216
Figure 5-154 TSSImportTranslationClasses (Class Diagram).....	5-219
Figure 5-155 DMSImportAcquireTask:execute (Sequence Diagram).....	5-222
Figure 5-156 DMSImportHandler:getCandidates (Sequence Diagram).....	5-223
Figure 5-157 DMSImportRitisAcquirer:connectIfNecessary (Sequence Diagram)	5-224
Figure 5-158 DMSImportRitisAcquirer:initialize (Sequence Diagram).....	5-226
Figure 5-159 DMSImportRitisAcquirer:onMessage (Sequence Diagram).....	5-227
Figure 5-160 EventImportModule:ExtSysConnStatusUpdate (Sequence Diagram)	5-228
Figure 5-161 EventImportRitisAcquirer:connectIfNecessary (Sequence Diagram).....	5-229
Figure 5-162 ExternalDeviceManagerImpl:searchCandidates (Sequence Diagram).....	5-230
Figure 5-163 ExternalDeviceManagerImpl:setCandidates (Sequence Diagram)	5-231
Figure 5-164 ExternalInterfaceModule:dmsTranslationStep1 Translate (Sequence Diagram)	5-232
Figure 5-165 ExternalInterfaceModule:handleDITranslationTask (Sequence Diagram)	5-233
Figure 5-166 ExternalInterfaceModule:handleDMSImportTask (Sequence Diagram)	5-234
Figure 5-167 ExternalInterfaceModule:handleExternalImport (Sequence Diagram).....	5-235
Figure 5-168 ExternalInterfaceModule:handleTITranslationTask (Sequence Diagram).....	5-236
Figure 5-169 ExternalInterfaceModule:handleTSSImportTask (Sequence Diagram).....	5-237
Figure 5-170 ExternalInterfaceModule:initializeDMSImportModule (Sequence Diagram)	5-238
Figure 5-171 ExternalInterfaceModule:initializeEventImportModule (Sequence Diagram)	5-239
Figure 5-172 ExternalInterfaceModule:initializeTSSImportModule (Sequence Diagram)	5-240
Figure 5-173 ExternalInterfaceModule:restartDMSImportModule (Sequence Diagram)	5-241
Figure 5-174 ExternalInterfaceModule:restartDMSImportModule (Sequence Diagram)	5-242
Figure 5-175 ExternalInterfaceModule:restartTSSImportModule (Sequence Diagram).....	5-243
Figure 5-176 ExternalInterfaceModule:shutdownDMSImportModule (Sequence Diagram)	5-244
Figure 5-177 ExternalInterfaceModule:shutdownTSSImportModule (Sequence Diagram)	5-245
Figure 5-178 ExternalInterfaceModule:tssTranslationStep1 Translate (Sequence Diagram).....	5-246
Figure 5-179 ExternalSystemConnectionImpl:init (Sequence Diagram)	5-248
Figure 5-180 ExternalSystemConnectionImpl:sendNotificationsIfNecessary (Sequence Diagram).....	5-249
Figure 5-181 TSSImportAcquireTask:execute (Sequence Diagram)	5-250
Figure 5-182 TSSImportRitisAcquirer:connectIfNecessary (Sequence Diagram)	5-252
Figure 5-183 TSSImportRitisAcquirer:initialize (Sequence Diagram)	5-253
Figure 5-184 TSSImportRitisAcquirer:onMessage (Sequence Diagram)	5-254
Figure 5-185 GeoAreaModulePkg (Class Diagram)	5-255
Figure 5-186 GeoAreaFactoryImpl:addGeoArea (Sequence Diagram).....	5-258
Figure 5-187 GeoAreaFactoryImpl:getGeoAreas (Sequence Diagram).....	5-259
Figure 5-188 GeoAreaFactoryImpl:removeGeoArea (Sequence Diagram)	5-260
Figure 5-189 GeoAreaFactoryImpl:updateGeoArea (Sequence Diagram).....	5-261
Figure 5-190 GeoAreaModulePkg:Initialize (Sequence Diagram).....	5-262
Figure 5-191 GeoAreaModulePkg:Shutdown (Sequence Diagram).....	5-263
Figure 5-192 HARControlModule (Class Diagram)	5-265
Figure 5-193 HARControlModule:SetConfiguration (Sequence Diagram)	5-274
Figure 5-194 HARControlModule:setConfigurationImpl (Sequence Diagram).....	5-276
Figure 5-195 INRIXDataImportModuleClasses (Class Diagram).....	5-277
Figure 5-196 CHART2.INRIXDataImportModule:DataImportTimerTask.run (Sequence Diagram)	5-283
Figure 5-197 CHART2.INRIXDataImportModule:INRIXDataImportModule.initialize (Sequence Diagram)	5-285
Figure 5-198 CHART2.INRIXDataImportModule:INRIXLinkDataProvider.getLinkData (Sequence Diagram)	5-286
Figure 5-199 CHART2.INRIXDataImportModule:PushINRIXLinkDataCmd.execute (Sequence Diagram).....	5-287

Figure 5-200 INRIXLinkDefImportProgram (Class Diagram)	5-288
Figure 5-201 INRIXDefLinkImportProgramPkg:importINRIXLinks	5-291
Figure 5-202 JavaClasses (Class Diagram)	5-292
Figure 5-203. MessageTemplateModule (Class Diagram)	5-297
Figure 5-204. DMSTravInfoMsgTemplateImpl:getConfig (Sequence Diagram)	5-300
Figure 5-205. DMSTravInfoMsgTemplateImpl:remove (Sequence Diagram)	5-301
Figure 5-206. DMSTravInfoMsgTemplateImpl:setConfig (Sequence Diagram)	5-302
Figure 5-207. MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate (Sequence Diagram)	5-303
Figure 5-208. MessageTemplateFactoryImpl:getDMSTravInfoMsgTemplates (Sequence Diagram)	5-304
Figure 5-209. MessageTemplateFactoryImpl:getTollRateTimeFormats (Sequence Diagram)	5-305
Figure 5-210. MessageTemplateModule:initialize (Sequence Diagram)	5-306
Figure 5-211. MessageTemplateModule:shutdown (Sequence Diagram)	5-307
Figure 5-212. RoadwayLocationModule (Class Diagram)	5-308
Figure 5-213. RoadwayLocation:ProvideCountyData (Sequence Diagram)	5-311
Figure 5-214. RoadwayLocationModule:Initialize (Sequence Diagram)	5-312
Figure 5-215. RoadwayLocationModule:Shutdown (Sequence Diagram)	5-313
Figure 5-216. SHAZAMControl (Class Diagram)	5-315
Figure 5-217. SHAZAMControlModule:setConfiguration (Sequence Diagram)	5-322
Figure 5-218. SHAZAMUtility (Class Diagram)	5-323
Figure 5-219. AlertManagement (Class Diagram)	5-324
Figure 5-220. Common (Class Diagram)	5-330
Figure 5-221. Common2 (Class Diagram)	5-335
Figure 5-222. DMSControl (Class Diagram)	5-339
Figure 5-223. DeviceManagement (Class Diagram)	5-348
Figure 5-224. ExternalDMS (Class Diagram)	5-352
Figure 5-225. ExternalSystem (Class Diagram)	5-356
Figure 5-226. FieldCommunications (Class Diagram)	5-359
Figure 5-227. GeoAreaManagement (Class Diagram)	5-363
Figure 5-228. HARControl (Class Diagram)	5-365
Figure 5-229. HARNotification (Class Diagram)	5-371
Figure 5-230. MessageTemplateManagement (Class Diagram)	5-374
Figure 5-231. ResourceManagement (Class Diagram)	5-377
Figure 5-232. TSSManagement (Class Diagram)	5-381
Figure 5-233. TrafficEventManagement (Class Diagram)	5-389
Figure 5-234. TrafficEventManagement2 (Class Diagram)	5-395
Figure 5-235. TrafficEventRule (Class Diagram)	5-400
Figure 5-236. TravelRouteManagement (Class Diagram)	5-403
Figure 5-237. TravelRouteManagement2 (Class Diagram)	5-411
Figure 5-238. UserManagement (Class Diagram)	5-414
Figure 5-239. RTMSObject (Class Diagram)	5-418
Figure 5-240. TSSManagementModulePkg (Class Diagram)	5-424
Figure 5-241. PolledTSSImpl:computeZoneGroupTrafficParms (Sequence Diagram)	5-431
Figure 5-242. PolledTSSImpl:processPollResults (Sequence Diagram)	5-432
Figure 5-243. PolledTSSImpl:setConfiguration (Sequence Diagram)	5-433
Figure 5-244. RTMSFactoryImpl:constructor (Sequence Diagram)	5-434
Figure 5-245. RTMSImpl:constructor (Sequence Diagram)	5-435
Figure 5-246. RTMSImpl:poll (Sequence Diagram)	5-436
Figure 5-247. TravelRouteModule (Class Diagram)	5-438
Figure 5-248. TravelRouteDB:getLinks (Sequence Diagram)	5-447
Figure 5-249. TravelRouteDB:getRoutes (Sequence Diagram)	5-448
Figure 5-250. TravelRouteFactoryImpl:addRoute (Sequence Diagram)	5-449
Figure 5-251. TravelRouteFactoryImpl:computeTravelTime (Sequence Diagram)	5-451
Figure 5-252. TravelRouteFactoryImpl:pushConsumerUpdate (Sequence Diagram)	5-453
Figure 5-253. TravelRouteFactoryImpl:sendUpdatesCompleted (Sequence Diagram)	5-454
Figure 5-254. TravelRouteFactoryImpl:updateLinkData (Sequence Diagram)	5-456
Figure 5-255. TravelRouteFactoryImpl:updateTollRateData (Sequence Diagram)	5-458

Figure 5-256. TravelRouteImpl:addRemoveConsumer (Sequence Diagram).....	5-459
Figure 5-257. TravelRouteImpl:computeTravelTime (Sequence Diagram).....	5-461
Figure 5-258. TravelRouteImpl:remove (Sequence Diagram)	5-462
Figure 5-259. TravelRouteImpl:setConfig (Sequence Diagram).....	5-464
Figure 5-260. TravelRouteImpl:setLinkStats (Sequence Diagram).....	5-465
Figure 5-261. TravelRouteImpl:setPartialConfig (Sequence Diagram)	5-466
Figure 5-262. TravelRouteModule:initialize (Sequence Diagram)	5-467
Figure 5-263. TravelRouteModule:shutdown (Sequence Diagram).....	5-468
Figure 5-264. UserManagementClassDiagram (Class Diagram).....	5-469
Figure 5-265. UtilityClasses (Class Diagram)	5-472
Figure 5-266. UtilityClasses2 (Class Diagram)	5-480
Figure 5-267. TravelTimeRange:constructor (Sequence Diagram).....	5-485
Figure 5-268. WrappersCD (Class Diagram)	5-487
Figure 5-269. WSTrafficEventExportModuleClasses (Class Diagram)	5-490
Figure 5-270. TrafficEventExportHandler:getTrafficEventList (Sequence Diagram)	5-494
Figure 5-271. TrafficEventHandler:initialize (Sequence Diagram).....	5-495
Figure 5-272. TrafficEventRequestHandler:processRequest (Sequence Diagram)	5-496
Figure 5-273. WSTrafficEventExportModule:initialize (Sequence Diagram)	5-497
Figure 5-274. WSTrafficEventExportModule:shutdown (Sequence Diagram).....	5-498
Figure 5-275. WSDMSExportModuleClasses (Class Diagram).....	5-500
Figure 5-276. DMSExportHandler:getDMSInventoryList (Sequence Diagram)	5-504
Figure 5-277. DMSExportHandler:getDMSStatusList (Sequence Diagram)	5-505
Figure 5-278. DMSExportHandler:initialize (Sequence Diagram)	5-506
Figure 5-279. DMSRequestHandler:handleExceptions (Sequence Diagram)	5-507
Figure 5-280. DMSRequestHandler:processRequest (Sequence Diagram).....	5-508
Figure 5-281. WSDMSExportModule:initialize (Sequence Diagram)	5-509
Figure 5-282. WSDMSExportModule:shutdown (Sequence Diagram)	5-510
Figure 5-283. TollRateImportModuleClasses (Class Diagram)	5-511
Figure 5-284. CHART2.webservices.tollrateimportmodule:TollDataManager.tollRateDataUpdated (Sequence Diagram).....	5-517
Figure 5-285. CHART2.webservices.tollrateimportmodule:TollDataManager.updateTollRateData (Sequence Diagram).....	5-518
Figure 5-286. CHART2.webservices.tollrateimportmodule:TollDataPushTask.run (Sequence Diagram)	5-519
Figure 5-287. CHART2.webservices.tollrateimportmodule:WSTollRateImportModule.initialize (Sequence Diagram).....	5-521
5-288. WebServicesBaseClasses (Class Diagram)	5-522
Figure 5-289. CHART2.webservices.base:BasicRequestHandler.processRequest (Sequence Diagram).....	5-528
Figure 5-290. CHART2.webservices.base:WebService.handleRequest (Sequence Diagram).....	5-529
Figure 5-291. CHART2.webservices.base:WebService.init (Sequence Diagram)	5-530
Figure 5-292. MiscDataClasses (Class Diagram)	5-531
Figure 5-293. MiscDataClasses2 (Class Diagram)	5-534
Figure 5-294. chartlite.data_location_classes (Class Diagram)	5-535
Figure 5-295. GUIDMSDataClasses (Class Diagram)	5-538
Figure 5-296. GUIDMSDataClasses2 (Class Diagram)	5-541
Figure 5-297. DiscoverDMSClassesCommand:discoverDMSClasses (Sequence Diagram)	5-542
Figure 5-298. DMSTravInfoMsgTrueDisplayMgr:updateGIF (Sequence Diagram)	5-543
Figure 5-299. WebChart2DMS:create (Sequence Diagram)	5-544
Figure 5-300. WebChart2DMS:setupDMSTravInfoMsgs (Sequence Diagram)	5-545
Figure 5-301. WebChart2DMS:updateConfig (Sequence Diagram)	5-546
Figure 5-302. WebChart2DMS:update_ModelChange (Sequence Diagram).....	5-547
Figure 5-303. WebDMSFactory:createDMS (Sequence Diagram)	5-548
Figure 5-304. GUIVideoDataClasses (Class Diagram)	5-549
Figure 5-305. GUILocationDataClasses (Class Diagram).....	5-552
Figure 5-306. GUIShazamClasses (Class Diagram).....	5-553
Figure 5-307. GUIHARDataClasses (Class Diagram)	5-555
Figure 5-308. chartlite.data.arbqueue_classes (Class Diagram)	5-559

Figure 5-309. GUITravelRouteClasses (Class Diagram)	5-561
Figure 5-310. chartlite.data.travelroutes.HistoryList:addElement (Sequence Diagram)	5-568
Figure 5-311. chartlite.data.travelroutes.HistoryList:toBucketArray (Sequence Diagram).....	5-569
Figure 5-312. chartlite.data.travelroutes.RoadwayLinkManager:addOrUpdateLink (Sequence Diagram).....	5-571
Figure 5-313. chartlite.data.travelroutes.RoadwayLinkManager:suggestLinks (Sequence Diagram).....	5-573
Figure 5-314. chartlite.data.travelroutes.TravelRoutePushConsumer:routeAdded (Sequence Diagram).....	5-574
Figure 5-315. chartlite.data.travelroutes.TravelRoutePushConsumer:routeConfigChanged (Sequence Diagram).....	5-575
Figure 5-316. chartlite.data.travelroutes.TravelRoutePushConsumer:routeDeleted (Sequence Diagram).....	5-576
Figure 5-317. chartlite.data.travelroutes.TravelRoutePushConsumer:routeTollRateUpdated (Sequence Diagram) ...	5-577
Figure 5-318. chartlite.data.travelRoutes.TravelRoutePushConsumer:routeTravelTimeUpdated (Sequence Diagram)	5-578
Figure 5-319. chartlite.data.travelroutes.WebTravelRoute:updateConfig (Sequence Diagram)	5-580
Figure 5-320. chartlite.data.travelroutes.WebTravelRoute:updateStatus (Sequence Diagram).....	5-582
Figure 5-321. chartlite.data.travelroutes.TravelRouteDiscovery (Sequence Diagram)	5-583
Figure 5-322. GUIMessageTemplateDataClasses (Class Diagram)	5-585
Figure 5-323. chartlite.data.templatemanagement.discoverTemplateClasses (Sequence Diagram).....	5-589
Figure 5-324. chartlite.data.templatemanagement.handleEventData (Sequence Diagram).....	5-590
Figure 5-325. GUIGeoAreaClasses (Class Diagram)	5-591
Figure 5-326. chartlite.data.geoareamgmt.discoverGeoAreaClasses (Sequence Diagram).....	5-593
Figure 5-327. data.alerts.classes (Class Diagram)	5-594
Figure 5-328. GUIExternalSystemClasses (Class Diagram)	5-598
Figure 5-329. chartlite.data.externalsystem:DiscoverExternalSystemClasses (Sequence Diagram).....	5-600
Figure 5-330. ExternalSystemPushConsumer:clientAdded (Sequence Diagram)	5-601
Figure 5-331. ExternalSystemPushConsumer:clientRemoved (Sequence Diagram)	5-602
Figure 5-332. ExternalSystemPushConsumer:clientUpdated (Sequence Diagram)	5-602
Figure 5-333. ExternalSystemPushConsumer:connectionStatusChanged (Sequence Diagram)	5-603
Figure 5-334. GUITSSDataClasses (Class Diagram)	5-604
Figure 5-335. chartlite.data.trafficevents_classes (Class Diagram).....	5-606
Figure 5-336. chartlite.data.trafficevents_event_type_classes (Class Diagram)	5-608
Figure 5-337. chartlite.data.trafficevents_misc_classes (Class Diagram)	5-610
Figure 5-338. plans_data_classes (Class Diagram)	5-612
Figure 5-339. plans_data_classes (Class Diagram)	5-614
Figure 5-340. ServletBaseClasses (Class Diagram).....	5-615
Figure 5-341. ServletMiscClasses (Class Diagram).....	5-617
Figure 5-342. DynImageCleanupTask:run (Sequence Diagram).....	5-619
Figure 5-343. chartlite.servlet.usermgmt.systemProfile_classes (Class Diagram)	5-620
Figure 5-344. SystemProfileReqHdlr:addDMSMsgComboProps (Sequence Diagram)	5-622
Figure 5-345. SystemProfileReqHdlr:getExternalConnectionAlertAndNotificationSettingsForm (Sequence Diagram)	5-623
Figure 5-346. SystemProfileReqHdlr:getExternalOrgToAgencyMappingsForm (Sequence Diagram).....	5-624
Figure 5-347. SystemProfileReqHdlr:getTSSSpeedSummaryRangesForm (Sequence Diagram)	5-625
Figure 5-348. SystemProfileReqHdlr:getTravelTimeMiscSettingsForm (Sequence Diagram)	5-626
Figure 5-349. SystemProfileReqHdlr:getTravelTimeRangesForm (Sequence Diagram).....	5-628
Figure 5-350. SystemProfileReqHdlr:getTravelTimeScheduleForm (Sequence Diagram).....	5-629
Figure 5-351. SystemProfileReqHdlr:setExternalAgencyToOrgMappings (Sequence Diagram).....	5-630
Figure 5-352. SystemProfileReqHdlr:setExternalConnectionAlertAndNotificationSettingsForm (Sequence Diagram)	5-631
Figure 5-353. SystemProfileReqHdlr:setTSSSpeedSummaryRanges (Sequence Diagram)	5-632
Figure 5-354. SystemProfileReqHdlr:setTravelTimeMiscSettings (Sequence Diagram).....	5-633
Figure 5-355. SystemProfileReqHdlr:setTravelTimeRanges (Sequence Diagram).....	5-635
Figure 5-356. SystemProfileReqHdlr:setTravelTimeSchedule (Sequence Diagram).....	5-637
Figure 5-357. chartlite.servlet.tss_classes (Class Diagram).....	5-638
Figure 5-358. chartlite.servlet.tss_dynlist_classes (Class Diagram).....	5-639
Figure 5-359. EditTSSLocationSupporter:setObjectLocation (Sequence Diagram)	5-643
Figure 5-360. TSSListSupporter:createDynList (Sequence Diagram)	5-645

Figure 5-361. TSSReqHdlr:getEditTSSLocationForm (Sequence Diagram)	5-646
Figure 5-362. chartlite.servlet.tss:setTSSConfigCommSettings (Sequence Diagram)	5-647
Figure 5-363. ServletDynListClasses (Class Diagram)	5-649
Figure 5-364. chartlite.servlet.dynlist.DynListReqHdlrDelegate:createDynList (Sequence Diagram)	5-652
Figure 5-365. chartlite.servlet.dynlist.DynListReqHdlrDelegate:setColumnVisibility (Sequence Diagram)	5-653
Figure 5-366. chartlite.servlet.dms.dynlist_classes (Class Diagram)	5-654
Figure 5-367. GUIDMSServletClasses (Class Diagram)	5-656
Figure 5-368. chartlite.servlet.dms:createDMSEditorData (Sequence Diagram)	5-659
Figure 5-369. chartlite.servlet.dms:createOrUpdateDMSTravInfoMsgTemplate (Sequence Diagram)	5-660
Figure 5-370. chartlite.servlet.dms:getAddEditDMSTravInfoMsgForm (Sequence Diagram)	5-661
Figure 5-371. chartlite.servlet.dms:getDMSEditorImageJSON (Sequence Diagram)	5-663
Figure 5-372. chartlite.servlet.dms:getDMSTravInfoMsgImageJSON (Sequence Diagram)	5-665
Figure 5-373. chartlite.servlet.dms:getDMSTravInfoMsgTemplateDataJSON (Sequence Diagram)	5-666
Figure 5-374. chartlite.servlet.dms:parseBasicConfigSettings (Sequence Diagram)	5-667
Figure 5-375. chartlite.servlet.dms:parseDMSTravInfoMsg (Sequence Diagram)	5-668
Figure 5-376. chartlite.servlet.dms:removeDMSTravInfoMsg (Sequence Diagram)	5-669
Figure 5-377. chartlite.servlet.dms:saveDMSEditorDataFromForm (Sequence Diagram)	5-670
Figure 5-378. chartlite.servlet.dms:setDMSConfigBasicSettings (Sequence Diagram)	5-671
Figure 5-379. chartlite.servlet.dms:setDMSTravelRoutes (Sequence Diagram)	5-672
Figure 5-380. chartlite.servlet.dms:setDMSTravelTimeDisplaySchedule (Sequence Diagram)	5-673
Figure 5-381. chartlite.servlet.dms:setDMSTravInfoMsgEnabledFlag (Sequence Diagram)	5-674
Figure 5-382. chartlite.servlet.dms:submitDMSTravInfoMsgForm (Sequence Diagram)	5-675
Figure 5-383. chartlite.servlet.dms:submitDMSTravInfoMsgTemplateForm (Sequence Diagram)	5-677
Figure 5-384. chartlite.servlet.dms:viewDMSMessageEditorForm (Sequence Diagram)	5-678
Figure 5-385. chartlite.servlet.dms:viewEditDMSTravelRoutesForm (Sequence Diagram)	5-679
Figure 5-386. DMSListSupporter:createDynList (Sequence Diagram)	5-680
Figure 5-387. DMSReqHdlr:getEditDMSLocationForm (Sequence Diagram)	5-681
Figure 5-388. EditDMSLocationSupporter:setObjectLocation (Sequence Diagram)	5-682
Figure 5-389. chartlite.servlet.alerts:resolveAlert (Sequence Diagram)	5-684
Figure 5-390. GUIGeographicAreasServletClasses (Class Diagram)	5-685
Figure 5-391. GeographicAreasReqHdlr:displayAddEditGeoAreaForm (Sequence Diagram)	5-686
Figure 5-392. GeographicAreasReqHdlr:importKMLFileJSON (Sequence Diagram)	5-687
Figure 5-393. GeographicAreasReqHdlr:removeGeoArea (Sequence Diagram)	5-688
Figure 5-394. GeographicAreasReqHdlr:submitAddEditGeoAreaForm (Sequence Diagram)	5-689
Figure 5-395. GUITravelRouteServletClasses (Class Diagram)	5-691
Figure 5-396. TravelRouteDynListSupporter:createDynList (Sequence Diagram)	5-696
Figure 5-397. TravelRouteDynListSupporter:getDynListSubjects (Sequence Diagram)	5-697
Figure 5-398. TravelRouteReqHdlr:addEditTravelRoute (Sequence Diagram)	5-698
Figure 5-399. TravelRouteReqHdlr:addEditTravelRouteForm (Sequence Diagram)	5-700
Figure 5-400. TravelRouteReqHdlr:addTravelRouteLink (Sequence Diagram)	5-701
Figure 5-401. TravelRouteReqHdlr:addTravelRouteLinkForm (Sequence Diagram)	5-703
Figure 5-402. TravelRouteReqHdlr:findTravelRouteLinksJSON (Sequence Diagram)	5-705
Figure 5-403. TravelRouteReqHdlr:getHistoryBucketTimes (Sequence Diagram)	5-706
Figure 5-404. TravelRouteReqHdlr:moveTravelRouteLink (Sequence Diagram)	5-707
Figure 5-405. TravelRouteReqHdlr:removeTollRateSource (Sequence Diagram)	5-708
Figure 5-406. TravelRouteReqHdlr:removeTravelRoute (Sequence Diagram)	5-709
Figure 5-407. TravelRouteReqHdlr:removeTravelRouteLink (Sequence Diagram)	5-710
Figure 5-408. TravelRouteReqHdlr:setTollRateSource (Sequence Diagram)	5-712
Figure 5-409. TravelRouteReqHdlr:setTollRateSourceForm (Sequence Diagram)	5-713
Figure 5-410. TravelRouteReqHdlr:setTravelRouteLinkSettings (Sequence Diagram)	5-715
Figure 5-411. TravelRouteReqHdlr:setTravelRouteLinkSettingsForm (Sequence Diagram)	5-717
Figure 5-412. TravelRouteReqHdlr:viewTravelRouteDetails (Sequence Diagram)	5-718
Figure 5-413. TravelRouteReqHdlr:viewTravelRouteLinkDetails (Sequence Diagram)	5-719
Figure 5-414. TravelRouteReqHdlr:viewTravelRoutes (Sequence Diagram)	5-721
Figure 5-415. GUIExternalSystemServletClasses (Class Diagram)	5-722
Figure 5-416. ExternalDeviceDynListSupporter:createDynList (Sequence Diagram)	5-725

Figure 5-417. ExternalDeviceDynListSupporter:getDynListSubjects (Sequence Diagram).....	5-726
Figure 5-418. ExternalSystemReqHdlr:addEditExternalClient (Sequence Diagram)	5-727
Figure 5-419. ExternalSystemReqHdlr:addEditTrafficEventInclusionRule (Sequence Diagram).....	5-728
Figure 5-420. ExternalSystemReqHdlr:downloadPrivateKey (Sequence Diagram)	5-729
Figure 5-421. ExternalSystemReqHdlr:generateKeyPair (Sequence Diagram)	5-730
Figure 5-422. ExternalSystemReqHdlr:getAddEditExternalClientForm (Sequence Diagram).....	5-731
Figure 5-423. ExternalSystemReqHdlr:getAddEditTrafficEventInclusionRuleForm (Sequence Diagram)	5-732
Figure 5-424. ExternalSystemReqHdlr:getExternalDeviceQueryForm (Sequence Diagram).....	5-733
Figure 5-425. ExternalSystemReqHdlr:removeExternalClient (Sequence Diagram).....	5-734
Figure 5-426. ExternalSystemReqHdlr:removeTrafficEventInclusionRule (Sequence Diagram)	5-735
Figure 5-427. ExternalSystemReqHdlr:submitExternalDeviceQueryForm (Sequence Diagram).....	5-736
Figure 5-428. ExternalSystemReqHdlr:submitExternalDeviceSelectionForm (Sequence Diagram)	5-737
Figure 5-429. ExternalSystemReqHdlr:viewExternalClientList (Sequence Diagram).....	5-738
Figure 5-430. ExternalSystemReqHdlr:viewExternalSystemConnectionStatus (Sequence Diagram).....	5-739
Figure 5-431. ExternalSystemReqHdlr:viewTrafficEventInclusionRules (Sequence Diagram)	5-740
Figure 5-432. DMSTravInfoMsgTemplateDynListClasses (Class Diagram).....	5-741
Figure 5-433. GUIMessageTemplateServletClasses (Class Diagram)	5-743
Figure 5-434. MessageTemplateReqHdlr:filterDMSTravInfoMsgTemplateList (Sequence Diagram)	5-745
Figure 5-435. MessageTemplateReqHdlr:getDMSTravInfoMsgTemplateList (Sequence Diagram)	5-746
Figure 5-436. MessageTemplateReqHdlr:removeDMSTravInfoMsgTemplate (Sequence Diagram)	5-747
Figure 5-437. MessageTemplateReqHdlr:sortDMSTravInfoMsgTemplateList (Sequence Diagram)	5-748
Figure 5-438. GUIVideoServletClasses (Class Diagram)	5-749
Figure 5-439. EditCameraLocationSupporter:setObjectLocation (Sequence Diagram).....	5-751
Figure 5-440. MonitorListSupporter:createDynList (Sequence Diagram)	5-752
Figure 5-441. VideoSourceConfigReqHdlr:getEditCameraLocationForm (Sequence Diagram).....	5-753
Figure 5-442. VideoSourceListSupporter:createDynList (Sequence Diagram).....	5-754
Figure 5-443. GUIDTrafficEventsDynListClasses (Class Diagram)	5-755
Figure 5-444. chartlite.servlet.trafficevents_classes (Class Diagram).....	5-758
Figure 5-445. AddTrafficEventReqHdlr:copyExternalEventAsCHARTEventWithoutForm (Sequence Diagram)....	5-761
Figure 5-446. TrafficEventDynListSupporter:addCol (Sequence Diagram)	5-762
Figure 5-447. TrafficEventDynListSupporter:createDynList (Sequence Diagram)	5-763
Figure 5-448. TrafficEventDynListSupporter:createDynListCols (Sequence Diagram).....	5-765
Figure 5-449. TrafficEventReqHdlr:getNearbyCameraJSONArray (Sequence Diagram)	5-766
Figure 5-450. TrafficEventReqHdlr:getNearbyDMSJSONArray (Sequence Diagram).....	5-768
Figure 5-451. TrafficEventReqHdlr:getNearbyDevicesJSON (Sequence Diagram).....	5-770
Figure 5-452. TrafficEventReqHdlr:getNearbyHARJSONArray (Sequence Diagram)	5-771
Figure 5-453. TrafficEventReqHdlr:getNearbyTSSJSONArray (Sequence Diagram)	5-774
Figure 5-454. TrafficEventReqHdlr:populateNearbyDeviceLocationJSON (Sequence Diagram).....	5-775
Figure 5-455. TrafficEventReqHdlr:setNearbyDevicesRadiusJSON (Sequence Diagram)	5-776
Figure 5-456. TrafficEventReqHdlr:viewEventDetails (Sequence Diagram)	5-778
Figure 5-457. TrafficEventXMLReqHdlr:getOpenTrafficEventsXML (Sequence Diagram).....	5-780
Figure 5-458. chartlite.servlet.location_classes (Class Diagram).....	5-781
Figure 5-459. SpecifyLocationReqHdlr:displayEditObjectLocationDataForm (Sequence Diagram).....	5-783
Figure 5-460. SpecifyLocationReqHdlr:getEditObjectLocationDataXML (Sequence Diagram)	5-784
Figure 5-461. SpecifyLocationReqHdlr:setObjectLocationDataXML (Sequence Diagram)	5-785
Figure 5-462. GUIHAZAMServletClasses (Class Diagram)	5-786
Figure 5-463. EditSHAZAMLocationSupporter:setObjectLocation (Sequence Diagram)	5-788
Figure 5-464. SHAZAMListSupporter:createDynList (Sequence Diagram)	5-789
Figure 5-465. SHAZAMReqHdlr:getAddSHAZAMForm (Sequence Diagram)	5-791
Figure 5-466. SHAZAMReqHdlr:getSHAZAMEditLocationForm (Sequence Diagram)	5-792
Figure 5-467. SHAZAMReqHdlr:processAddSHAZAM (Sequence Diagram).....	5-794
Figure 5-468. GUIHARServletClasses (Class Diagram).....	5-795
Figure 5-469. EditHARLocationSupporter:setObjectLocation (Sequence Diagram).....	5-798
Figure 5-470. HARListSupporter:createDynList (Sequence Diagram)	5-799
Figure 5-471. HARReqHdlr:getEditHARLocationForm (Sequence Diagram)	5-801

Figure 5-472. GUIFlexComponentsClasses (Class Diagram)	5-802
Figure 5-473. GUIFlexEditLocationClasses (Class Diagram).....	5-803
Figure 5-474. chartlite.util_classes (Class Diagram).....	5-805
Figure 5-475. ServletUtil:getNearbyObjects (Sequence Diagram)	5-807
Figure 5-476. DynamicListClasses (Class Diagram).....	5-808

1 Introduction

1.1 Purpose

This document describes the design of the software for Release 3, Build 3 of the CHART system. This build provides:

- **Traveler Information Messages**, including Travel Times and Toll Rates. The system will allow automatic display of DMS messages that include travel times and/or toll rates for one or more destinations. Data for these messages will be supplied by CHART travel routes, new for R3B3. Features to allow travel routes to be added, configured, viewed, and removed are included in R3B3. The underlying travel time data for travel routes will be obtained via an external connection to the INRIX system, while underlying toll rate data for travel routes will be provided to CHART by the Vector system. R3B3 will provide consistent formats for traveler information messages using system-wide message templates. Traveler information message templates can be configured by an administrator to specify the layout of a message as well as the message content, including data provided by CHART travel routes.
- **Device Locations**. R3B3 adds location fields to CHART devices, including DMS, TSS (Detectors), HAR, SHAZAM, and Cameras. In device lists, users can sort and filter on location fields, making it easier to view devices by county, roadway, or even the order they occur on the roadway (by mile marker). The addition of device locations in R3B3 also allows the system to show devices close to traffic events to aid in event response.
- **External Event Import Enhancements**. New R3B3 features allow rules to be defined to control which external events get imported from RITIS into CHART. These rules contain various criteria that external events must meet for CHART to import them. Each rule can also specify if external events meeting that rule should generate an External Event Alert (new for R3B3). Likewise, rules can specify if external events that meet the rule criteria should be automatically marked as “interesting” and be shown on the CHART system home page. To support geographical criteria in these import rules, capability to manage geographical area definitions is included in R3B3. These geographical areas can be used in external event import rules and can also be used when managing external DMS and TSS to be included in CHART (see below), and lay the foundation for inclusion of Areas of Responsibility in a future release.
- **External DMS and TSS**. R3B3 adds the ability to import DMS and TSS devices from RITIS. The administrator can manage the potentially large list of external devices available to pick and choose which devices will be included in CHART. Once included in CHART, the status of external devices will be maintained in the CHART system, allowing users to see current DMS messages and current TSS traffic parameters (volume, speed, occupancy). These external devices will be included when showing users devices close to traffic events.
- **Web Service to Provide CHART Data to External Systems**. A new web service is included in R3B3 to allow external systems such as RITIS to connect to CHART to

retrieve information. This new service provides better security and data protection than the existing CORBA based interface. It also provides better isolation of the CHART system from external systems.

- Enhanced user rights for Traffic Events, Detectors, and Device Configuration data. New user rights are included in R3B3 to allow better protection at viewing sensitive traffic event data, such as the indication that a fatality is involved. New user rights for detectors allow volume, speed, and occupancy (VSO) data to be better protected. Users can be given the right to view detailed VSO data or summary VSO data (speed range) based on the owning organization of each device. The user rights for device configuration data are changed in R3B3 to make them consistent across device types and to ensure sensitive settings are only shown to permitted users.
- TCP/IP Communications for DMS and TSS. The R3B3 system allows DMS and TSS devices to be connected to CHART via TCP/IP communications. A new option is added to the DMS and TSS configuration pages to allow this new communication option to be used.
- NTCIP DMS Font Settings. New configuration settings for NTCIP DMSs are added in R3B3 to allow the default font and line spacing to be set and will be used by CHART each time a message is set. This works around a problem where the default font on NTCIP DMSs gets reset when power is lost.
- Miscellaneous Enhancements. Several other enhancements are added to CHART in R3B3:
 - Show / Display Columns in Lists. When viewing a list of devices, traffic events, or travel routes in the working window, the user can choose which columns to show. Several new columns are being added to these lists, such as location fields, and this new feature allows the user to keep their lists at a minimum width by displaying only the columns in which they are interested.
 - Show Devices Close to an Event. A new section is added to the traffic event details page in R3B3 to show devices located within a specified radius of the traffic event. The radius can be changed by the user per traffic event, and the user can choose to add close DMSs and HARs directly to the event's response plan.
 - Lane Level Detector Data. A new feature in R3B3 allows users to view the volume, speed, and occupancy data for each detection zone within a detector's zone groups. This data is also being made available to external systems such as RITIS (with appropriate user rights).
 - DMS Alerts and Notifications. New DMS configuration values in R3B3 allow the device failure alert to be sent for DMS communications failures in addition to hardware failures. Notifications can also be configured to be sent when a hardware failure or communication failure is detected. These settings are configured separately for each DMS.

1.2 Objectives

The main objective of this detailed design document is to provide software developers with a framework in which to implement the requirements identified in the CHART R3B3 Requirements document. A matrix mapping requirements to the design is presented in Section 6.

1.3 Scope

This design is limited to Release 3, Build 3 (R3B3) of the CHART System. It addresses both the design of the server components of CHART and the Graphical User Interface (GUI) components of CHART. Since the CHART GUI is browser based, the GUI refers to both the user interface and the components actually executing on the web server. This design does not include designs for components implemented in earlier releases of the CHART system.

1.4 Design Process

The design was created by capturing the requirements of the system in UML Use Case diagrams. Class diagrams were generated showing the high level objects that address the Use Cases. Sequence diagrams were generated to show how each piece of major functionality will be achieved. This process was iterative in nature – the creation of sequence diagrams sometimes caused re-engineering of the class diagrams, and vice versa.

1.5 Design Tools

The work products contained within this design will be extracted from the Tau Unified Modeling Language (UML) Suite design tool. Within this tool, the design will be contained in the CHART project, CHART R3B3, Analysis phase and System Design phase.

1.6 Work Products

The final R3B3 design consists of the following work products:

- Use Case diagrams that capture the requirements of the system
- UML Class diagrams, showing the software objects which allow the system to accommodate the uses of the system described in the Use Case diagrams
- UML Sequence diagrams showing how the classes interact to accomplish major functions of the system

2 Architecture

The sections below discuss specific elements of the architecture and software components that are created, changed, or used in R3B2.

2.1 Network/Hardware

CHART R3B3 will introduce several new interfaces for CHART – the INRIX system for travel times, the Vector system for toll rates, and additional RITIS connections to import DMS and TSS devices. Additionally, an export service is included to allow external systems to retrieve data from the CHART system. CHART will connect to the INRIX system using its HTTPS/XML interface. Vector will connect to the CHART system on an HTTPS/XML interface hosted on the CHART system. The CHART system will also host an HTTPS/XML interface for data export. The connections to RITIS to import DMS and TSS devices will use the same type of connection that is used in R3B2 for importing traffic events from RITIS: a JMS connection initiated from a CHART service.

2.2 Software

CHART uses the Common Object Request Broker Architecture (CORBA) as the base architecture, with custom built software objects made available on the network to allow their data to be accessed via well defined CORBA interfaces. Communications to remote devices use the Field Management Server (FMS) architecture. This architecture will continue forward for Release R3B3. There will be no major changes to the CHART software architecture infrastructure.

2.2.1 COTS Products

CHART uses numerous COTS products for both run-time and development.

Product Name	Description
Apache ActiveMQ	CHART uses this to connect to RITIS JMS queues
Apache Jakarta Ant	CHART uses Apache Jakarta Ant 1.6.5 to build CHART applications and deployment jars.
Apache Tomcat	CHART R3B3 will use Apache Tomcat 6.0.18 as the GUI web server.
Attention! CC	CHART uses Attention! CC Version 2.1 to provide notification services.
Attention! CC API	CHART uses Attention! CC API Version 2.1 to interface with Attention! CC.
Attention! NS	CHART uses Attention! NS Version 6.1 to provide notification services.
Bison/Flex	CHART uses Bison and Flex as part of the process of compiling binary macro files used for performing camera menu operations on Vicon Surveyor VFT cameras.

CoreTec Decoder Control	CHART uses a CoreTec supplied decoder control API for commanding CoreTec decoders.
Dialogic API	CHART uses the Dialogic API for sending and receiving Dual Tone Multi Frequency (DTMF) tones for HAR communications.
Flex2 SDK	The R3B3 CHART GUI will use the Flex2 SDK, version 3.1 to provide the Flex compiler, the standard Flex libraries, and examples for building Flex applications.
GIF89 Encoder	Utility classes that can create .gif files with optional animation. This utility is used for the creation of DMS True Display windows.
JDOM	CHART uses JDOM b7 (beta-7) dated 2001-07-07. JDOM provides a way to represent an XML document for easy and efficient reading, manipulation, and writing.
JacORB	CHART uses a compiled, patched version of JacORB 2.2.4. The JacORB source code, including the patched code, is kept in the CHART source repository.
Java Run-Time (JRE)	CHART R3B3 will use 1.5.0_16.
JavaService	CHART uses JavaService to install the server side Java software components as Windows services.
JAXEN	CHART uses JAXEN 1.0-beta-8 dated 2002-01-09. The Jaxen project is a Java XPath Engine. Jaxen is a universal object model walker, capable of evaluating XPath expressions across multiple models.
JoeSNMP	CHART uses JoeSNMP version 0.2.6 dated 2001-11-11. JoeSNMP is a Java based implementation of the SNMP protocol. CHART uses for commanding iMPath MPEG-2 decoders and for communications with NTCIP DMSs.
JSON-simple	CHART uses the JSON-simple java library to encode/decode strings that use JSON (JavaScript Object Notation).
JTS	CHART uses the Java Topology Suite (JTS) version 1.8.0 for geographical utility classes.
NSIS	CHART uses the Nullsoft Scriptable Installation System (NSIS), version 2.20, as the server side installation package.
Nuance Text To Speech	For text-to-speech (TTS) conversion CHART uses a TTS engine that integrates with Microsoft Speech Application Programming Interface (MSSAPI), version

	5.1. CHART uses Nuance Vocalizer 4.0 with Nuance SAPI 5.1 Integration for Nuance Vocalizer 4.0.
Oracle	CHART uses Oracle 10.1.0.5 as its database and uses the Oracle 10G JDBC libraries (ojdbc1.4.jar) for all database transactions.
O'Reilly Servlet	Provides classes that allow the CHART GUI to handle file uploads via multi-part form submission.
Prototype Javascript Library	The CHART GUI uses the Prototype Javascript library, version 1.5.1, a cross-browser compatible Javascript library provides many features (including easy Ajax support).
SAXPath	CHART uses SAXPath 1.0-beta-6 dated 2001-09-27. SAXPath is an event-based API for XPath parsers, that is, for parsers which parse XPath expressions.
Velocity Template Engine	Provides classes that CHART GUI uses in order to create dynamic web pages using velocity templates.
Vicon V1500 API	CHART uses a Vicon supplied API for commanding the ViconV1500 CPU to switch video on the Vicon V1500 switch

2.2.2 Deployment /Interface Compatibility

2.2.2.1 External Interfaces

The diagram below presents an overall view of CHART within the context of other external systems. The green boundaries represent devices that the CHART software communicates with directly. The major external interfaces include:

1. CHART Web Server – Receives information from the CHART system for publishing on the Web. This information includes incident reports, lane closure data, speed sensor data, DMS messages, and camera video. In the future (sometime after R3B3 is deployed), the CHART Web Server will migrate to using the CHART HTTPS/XML external interface to obtain data from the CHART system.
2. CHART Map – The CHART Web Event Listener is used to receive CORBA Events from CHART relating to roadway conditions for display with the CHART Mapping application. The data includes incident reports, lane closure data, DMS messages, and speed sensor data. CHART also queries the mapping database to get counties, roads, and road intersection data. In the future (sometime after R3B3 is deployed), the CHART Map will migrate to using the CHART HTTPS/XML external interface to obtain data from the CHART system.
3. Emergency Operations Reporting System (EORS) – Legacy system providing information on road closures and road status.
4. Media – Commercial and public broadcasters.
5. SCAN – SHA legacy system supplying weather sensor data.

6. CHART Reporting Tool – Generates reports from data on CHART databases.
7. University of Maryland Center for Advanced Transportation Technology (CATT) Lab as Regional Integrated Transportation Information System (RITIS) - Receives CORBA Events from CHART and will migrate to use the CHART HTTPS/XML external interface for this purpose in the future (sometime after R3B3 is deployed). Provides SAE J2354 standard regional traffic events and TMDD standard DMS and TSS data via java messaging service connections.
8. Notification Recipients – Receive notification from CHART about significant events via e-mail or page/text.
9. INRIX – External system that provides travel time data to the CHART system. CHART connects to INRIX via an HTTPS/XML interface.
10. Vector – External (MdTA) system that provides toll rate data to the CHART system. The Vector system connects to CHART via an HTTPS/XML interface provided by CHART.

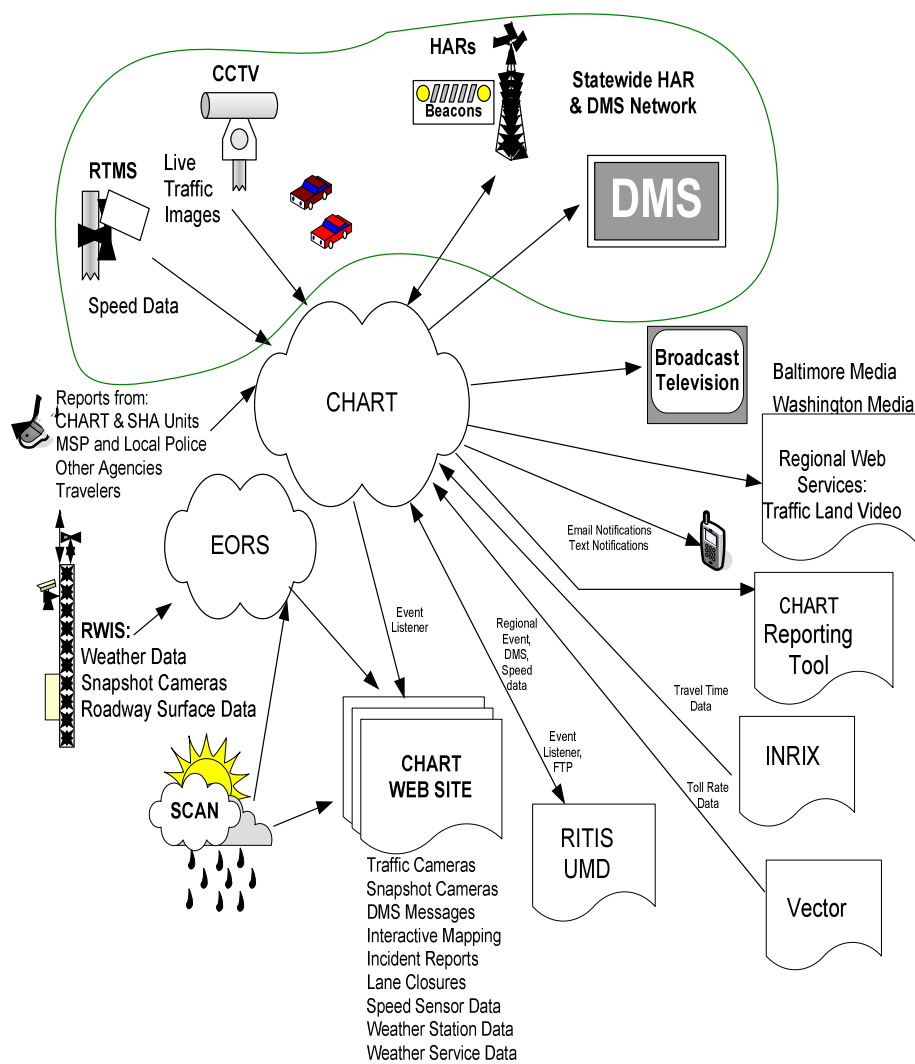


Figure 2-1 CHART and External Interfaces

For CHART R3B3, the following diagram describes how the new external interfaces will be deployed within CHART.

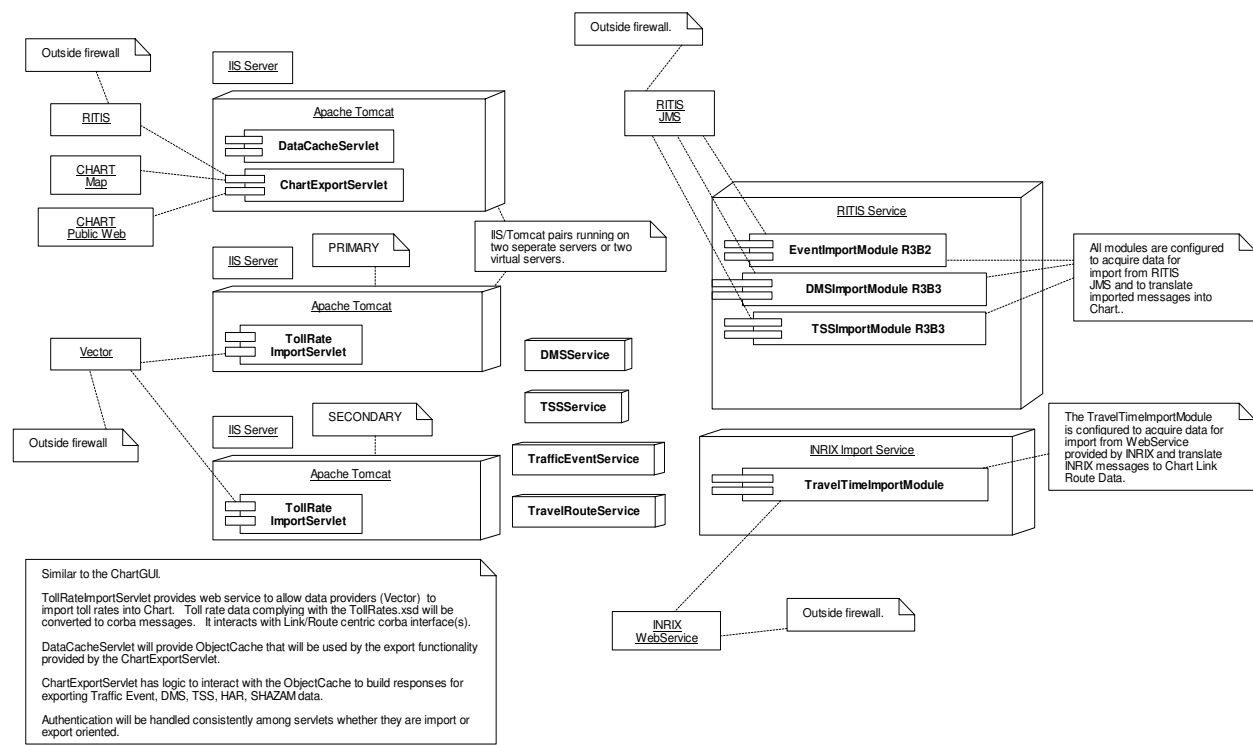


Figure 2-2 CHART R3B3 External Interface Deployment

2.2.2.2 Internal Interfaces

The architecture for the CHART system distributes complete system functionality to a number of districts throughout the State of Maryland. Each of these complete systems can provide full functionality for the devices connected to the system and objects created within that system (such as traffic events), and provides functionality for other district's systems that are available. Thus the absence of one district's server does not affect the ability of another district to use their own system or other systems that are available. Although the server deployment is spread across multiple sites, the user sees one large system, as CORBA is used to pull together objects served from the many deployment sites.

The CHART GUI is able to locate the software objects at all deployment sites through the use of the CORBA Trading Service. A CORBA Trading Service runs at each deployment site. Each CHART service that publishes CORBA objects offers the objects through its local CORBA Trading Service. The GUI provides a unified view of the system, even though the system is actually distributed over multiple deployment sites.

In addition to showing the software objects throughout the system on a single interface, it is also necessary to reflect the current state of the software objects as they are changed during real time

operations. The CORBA Event Service is used to allow objects to push changes in their state to the GUI, other back end CHART services, the CHART Event Listener, or any other interested CORBA clients. Each deployment site has an instance of a CORBA Event Channel Factory, which is an extension of the CORBA Event Service that allows multiple event channels. Each CHART service whose objects are subject to real time changes will create one or more Event Channels in its local Event Channel Factory. Each event channel is earmarked for a specific class of events (such as DMS events). Each service that creates channels in the CORBA Event Channel Factory publishes the event channel in the CORBA Trading Service and then uses the channel to push events relating to object state, configuration updates, etc.

An interface that wishes to listen for events at a system wide level discovers all of the event channels via the CORBA Trading Service and registers itself as a consumer on each of the event channels. Using this scheme, an interface uses the Trading Service to discover all software objects and Event Channels regardless of their deployment site. The interface may then provide the user with a unified view of the system, both in the objects presented and the ability to show near real time updates of these objects. Since the nature of the system is dynamic, processes periodically rediscover new objects and event channels from known districts via the Trading Service.

Most CHART background services which communicate with physical devices deployed along Maryland highways do so via FMS servers. One or more CHART Communications Services run on each FMS in the system. The CHART background services requiring FMS services for this purpose are the DMS Service, HAR Service (which also serves SHAZAMs), and the TSS Service. The communications between these three services and the Communications Services are IIOP, over TCP/IP. Communications from the Communications Services out to the physical devices are accomplished by telephone (via either POTS or ISDN modems, or via Telephony DTMF communications) or by direct serial connection. Telephone service is usually provided via landline, although cellular service occasionally needs to be utilized.

CHART background services that communicate with physical DMS and TSS devices also allow direct TCP/IP communications if supported by the devices. FMS servers and CHART Communications Services are not used by these background services to communicate with devices configured for this type of communication.

The remaining CHART background service controlling physical field devices is the Video Service. Video communication is accomplished via TCP/IP. Communication to CoreTec decoders is accomplished via proprietary CoreTec protocol over TCP/IP. Communication to iMPath decoders is accomplished via SNMP over TCP/IP, with published MIBs. CHART does not directly command either the iMPath or the CoreTec encoders; they are used only as a pass-through to pass camera control commands and responses to/from the attached cameras. CHART's communication with the encoders, then, is via TCP/IP with no proprietary protocol involved. Communications to the Vicon V1500 NTSC video switch is accomplished via a proprietary Vicon protocol over TCP/IP. Once video connections are thus established, video flows directly from encoder to decoder via MPEG2 or MPEG4 over TCP/IP, and/or through a V1500 analog video switch.

The following deployment diagrams show the deployment of CHART at a single district within the larger CHART system. The diagrams depict the various computers that are deployed at the site. Each computer shows the processes that are installed and running on it. The lines between the computers show the protocols that are used for communication between the various processes involved. The GUI deployment diagram shows that the web browser (Internet Explorer) on the operator workstation can send requests to the GUI web server machine using the standard HTTP or HTTPS protocols. These requests are handled by the Microsoft IIS web server process which uses the requested URL to determine that the request is intended for the CHART GUI servlet application. IIS forwards requests for CHART to the installed Apache Tomcat application which passes the request to the CHART GUI Servlet for processing. This servlet communicates with the processes on the CHART Server machine via the standard CORBA IIOP protocol which utilizes the TCP/IP protocol. Additionally this servlet communicates with the CHART Database server via the JDBC API which utilizes the TCP/IP protocol.

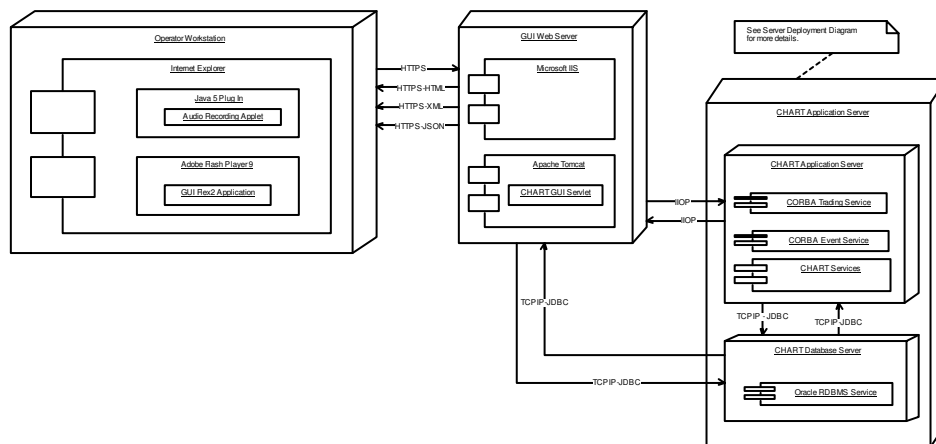


Figure 2-3 CHART Internal Interfaces (GUI Deployment)

The server deployment diagram shows the services running on the CHART application server in more detail. New for R3B3 are the Travel Route Service, the INRIX import service, and an IIS/Tomcat instance used to allow Vector to supply toll rate data and to allow CHART to provide its data to external systems. The CHART application server uses the standard CORBA IIOP protocol to communicate to the GUI web server to handle user requests and to update system state, and to the field management (FMS) server to communicate to DMS, HAR, SHAZAM, and TSS field devices. It also uses TCP/IP to control camera and monitor video devices and certain DMS and TSS devices. Finally, the CHART application server communicates with the CHART Mapping database to obtain roadway location information via the JDBC API which utilizes the TCP/IP protocol.

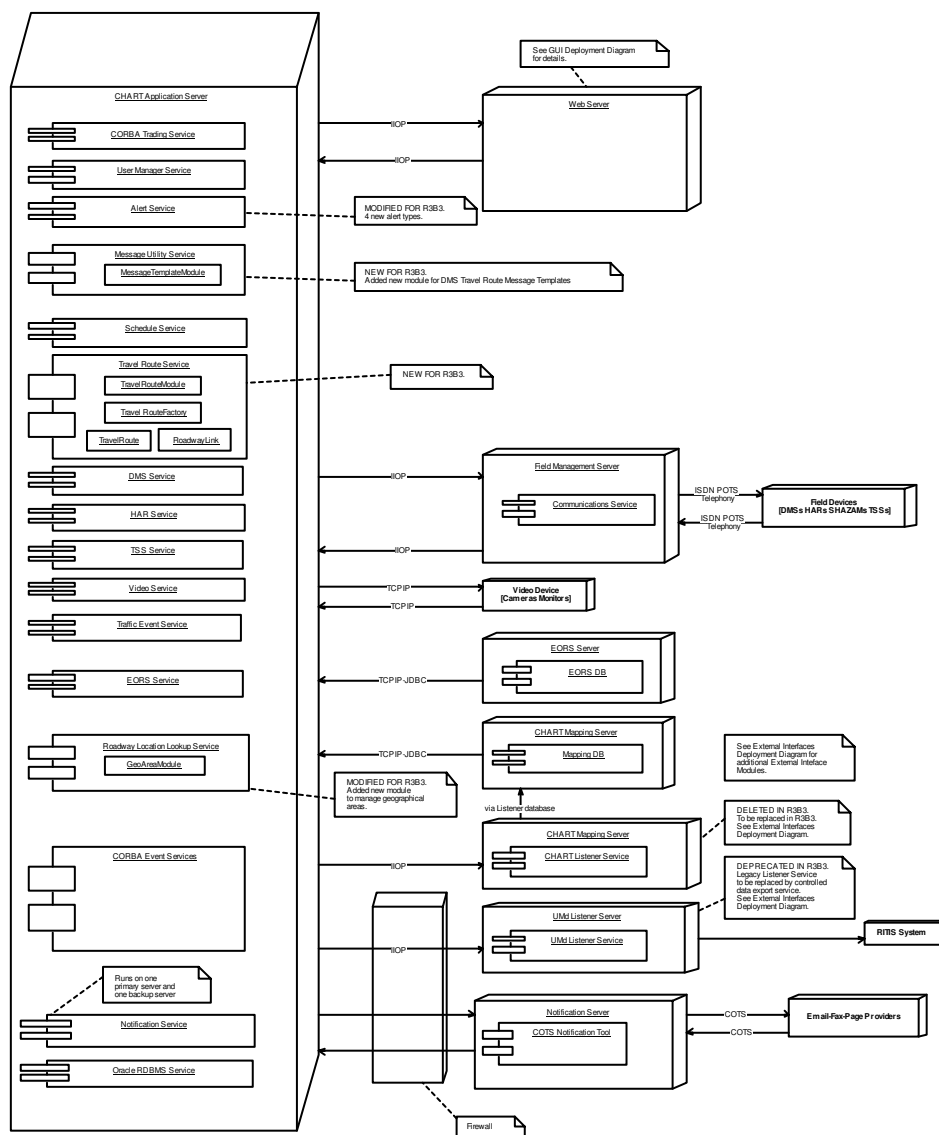


Figure 2-4 CHART Internal Interfaces (Server Deployment)

2.3 Security

The CHART System itself runs entirely behind the MDOT firewall. CHART R3B3 requires a new interface to the INRIX system which resides on the Internet. CHART connects to INRIX via HTTPS. This connection requires an outbound connection through the MDOT firewall to port 443 of the INRIX server. This is permitted because the connection is initiated from within the MDOT network.

CHART R3B3 introduces two new connections to the RITS system developed by the University of Maryland, making a total of 3 (one connection already exists as of R3B2). These connections require the opening of a few specific ports in the MDOT firewall. This is permitted because the connection is initiated from within the MDOT network.

Since the CHART System runs entirely behind the MDOT firewall, user access to the CHART system via the GUI from the outside world must be specifically enabled for users to connect from specific external locations. Control of video cameras is ostensibly limited to users which can see camera images on a local monitor, which are limited in number and restricted to controlled locations within designated facilities.

The CHART browser interface can be configured to run with HTTP or HTTPS (Secure HTTP). The fielded production system is always configured to run with HTTPS. HTTPS provides an additional SSL or TLS encryption/authentication layer between HTTP and TCP, which protects data in transit between the client machine web browser and the web server machine. Additionally, the system runs with Microsoft's Internet Information Services (IIS).

All users connecting to CHART are required to provide a user name and password before any CHART information is provided or any actions can be attempted. Invalid login attempts are logged to the CHART Operations Log (database table), a permanently archived log of system activity. Users with appropriate rights can see all users logged into the system and can force users off the system at any time, directly from the CHART GUI. Before editing the CHART dictionary, a particularly sensitive area, a logged on user is reauthenticated on the spot by requiring the user to provide a user name and password again.

When a legitimate CHART user logs in, he or she is granted certain functional rights, based on the user ID. These rights typically include, for instance, the ability to create, edit and close traffic events and create and execute response plan items in response to traffic events. Other rights allow direct interaction with CHART devices, such as the ability to put them offline, online, or into maintenance mode, and to issue maintenance mode commands. Video rights are very granular, so camera control rights can be issued with a very fine grain. Users cannot perform actions for which they do not have rights. Typically rather than graying out buttons, prohibited actions do not even appear on the user's browser, so in most cases users may not even know what they are missing. There is a special "view-only" user configured which can see CHART status within the system but cannot perform any actions which would change system status in any way.

Rights can be assigned to users on an organization-by-organization level. For instance, a user may be able to issue maintenance commands on one organization's DMSs, but not others. The rights are stored in an opaque access control token obtained during the login transaction. Users cannot see or modify this token, and generally are not aware of its existence. It is held by the web service on behalf of the user and is passed from the web service to the background services on all but the most benign service requests.

CHART R3B3 introduces a web service that allows the MdTA Vector system to provide toll rate data to the CHART system. This data will be provided over HTTPS on a non-standard port (not 443). A second web service introduced in R3B3 allows external systems (such as RITIS) to obtain data from CHART. This will eventually eliminate the need for CHART to allow external systems to connect via its CORBA interfaces, although access to the CORBA interfaces will still be required until existing external consumers of CHART data are updated to use the new HTTPS/XML interface. These new web services will require the opening of a specific port in

the MDOT firewall. This is permitted because these services use a non-standard port for HTTPS.

All external systems that connect to a CHART HTTPS/XML web service (Vector, to supply toll rate data, and others to obtain data from CHART) will be assigned a unique client ID and must be pre-configured in the CHART system by an Administrator to allow access. A public/private key pair will be generated by the Administrator for each external system, with the public key being stored in the CHART system, and the private key being provided to the external system owner for their use when connecting to the CHART system. Each request received by an external system will include the external system client ID and a digital signature created with their private key. CHART will validate all requests using the client's public key to ensure the request is from a trusted source. The Vector system provides data to CHART and does not request data from CHART, and for this reason the signature validation is all that is required before CHART accepts toll rate data from Vector. For external systems retrieving data from CHART via the HTTPS/XML interface, each external system client ID will also be pre-configured in CHART by an Administrator to assign one or more CHART user roles. The CHART user roles and the functional rights contained in each role will be used by CHART to determine the data an external client is permitted to retrieve from CHART, and in some cases the detail of the data retrieved. (For example for some detectors an external system may be provided actual speeds, others a speed range, and yet others no speed data, depending on the organization that owns the detector and the functional rights assigned to the client's role(s)).

2.4 Data

CHART R3B3 will be tested with the Oracle database patches that are available and will be deployed in the field at the time of CHART R3B3 deployment. The database patches may possibly be applied in the field before CHART R3B3 deployment.

2.4.1 Data Storage

The CHART System stores most of its data in an Oracle database. However, some data is stored in flat files on the CHART servers. This section describes both types of data.

2.4.1.1 Database

2.4.1.1.1 Database Architecture

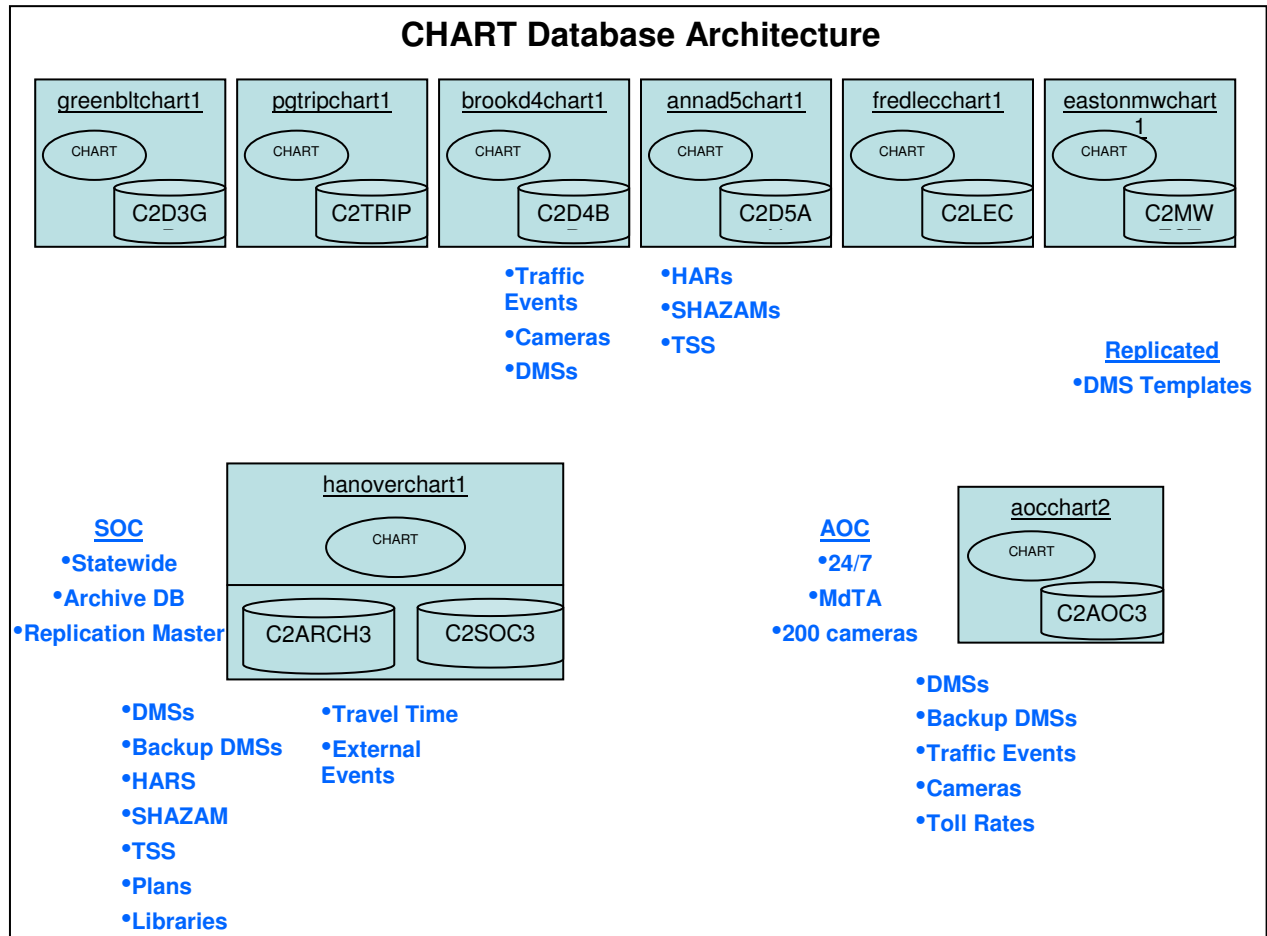
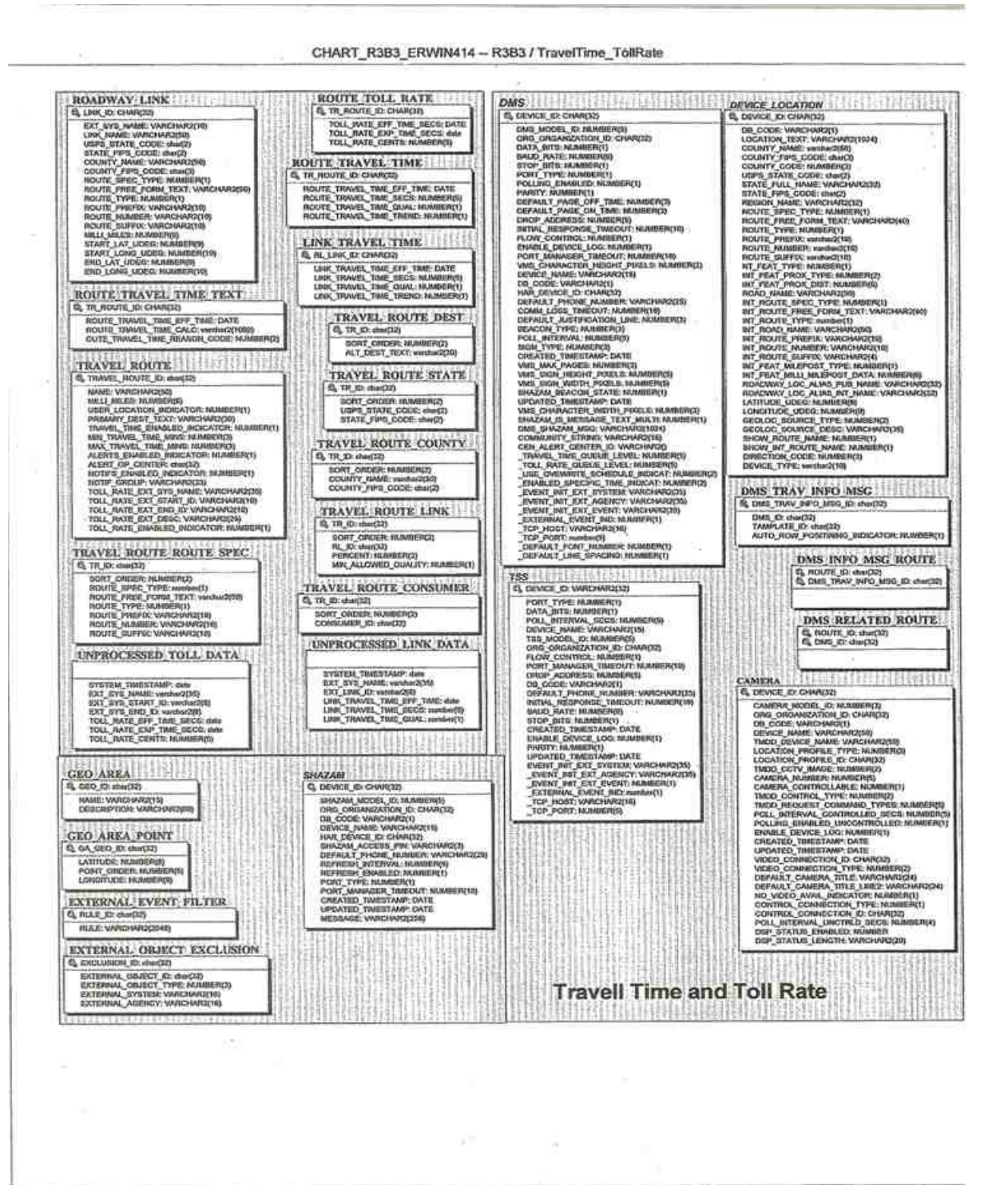


Figure 2-5 CHART R3B3 Database Architecture

2.4.1.1.2 Logical Design

2.4.1.1.2.1 Entity Relationship Diagram (ERD)



Travell Time and Toll Rate

CHART_R3B3_ERWIN414 – R3B3 / R3B3_Video

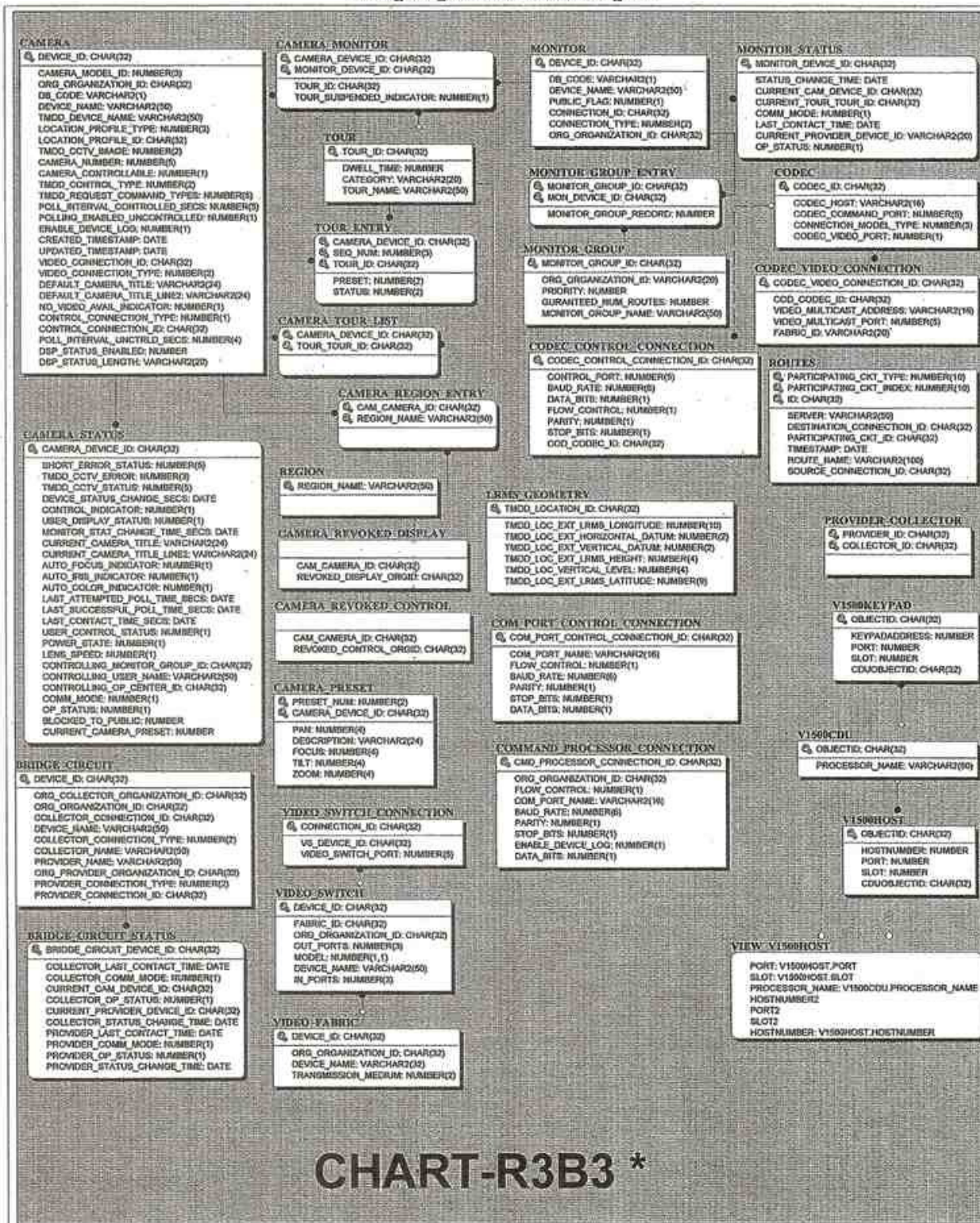


CHART-R3B3 *

1,1 / 1,1 – 11:47:24 AM, 12/17/2008

HAR

DEVICE_ID: CHAR(32)
DEFAULT_MONITOR_PHONE_NUMBER: VARCHAR(255)
MONITOR_PORT_TYPE: NUMBER(1)
MAX_TIME: NUMBER(5)
DEVICE_NAME: VARCHAR(15)
DEVICE_LOCATION: VARCHAR(50)
HAR_MODEL_ID: NUMBER(5)
ORG_ORGANIZATION_ID: CHAR(32)
MONITOR_PORT_MANAGER_TIMEOUT: NUMBER(10)
PORT_TYPE: NUMBER(1)
HAR_ACCESS_PIC: VARCHAR(255)
DEFAULT_PHONE_NUMBER: VARCHAR(255)
DB_CODE: VARCHAR(1)
PORT_MANAGER_TIMEOUT: NUMBER(10)
DEFAULT_BODY_CLIP_PIC: NUMBER(20)
CREATED_TIMESTAMP: DATE
DEFAULT_TRAILER_CLIP_PIC: NUMBER(20)
DEFAULT_HEADER_CLIP_PIC: NUMBER(20)
UPDATED_TIMESTAMP: DATE
ENABLE_DEVICE_LOG: VARCHAR(10)
MASTER_HAR_ID: VARCHAR(32)

HAR_PHONE_NUMBER

PORT_MANAGER_NAME: VARCHAR(255)
PHONE_NUMBER: VARCHAR(255)
PORT_MANAGER_TIMEOUT: NUMBER(10)
HAR_DEVICE_ID: CHAR(32)
MONITOR_INDICATOR: NUMBER(1)
DB_CODE: VARCHAR(1)

HAR_SLOT_CONFIG

HAR_DEVICE_ID: CHAR(32)
SLOT_NUMBER: NUMBER(3)
SLOT_USAGE_CODE: NUMBER(3)
DB_CODE: VARCHAR(1)
HMC_HAR_CLIP_PIC: NUMBER(20)

HAR STATUS

HAR_DEVICE_ID: CHAR(32)
STATUS_CHANGE_TIME: DATE
LAST_CONTACT_TIME: DATE
CEN_CENTER_ID: CHAR(32)
DEVICE_STATE_CODE: NUMBER(3)
COMM_STATUS: NUMBER(1)
HAR_INITIALIZED: NUMBER(1)
TRANSMITTER_STATE: NUMBER(1)
LAST_TIMESTAMP_REFRESH_TIME: DATE
HMC_HAR_MSG_PIC: NUMBER(10)

HAR NOTIFIER

HAR_DEVICE_ID: CHAR(32)
HAR_DEVICE_ID: CHAR(32)
ACTIVE_INDICATOR: NUMBER(1)
DB_CODE: VARCHAR(1)
DEVICE_CODE: NUMBER(3)

HAR_CLIP_AUDIO

HAR_CLIP_AUDIO_ID: CHAR(32)
HAR_AUDIO: BLOB
DB_CODE: VARCHAR(1)

HAR AUDIO OWNER

HCA_HAR_CLIP_AUDIO_ID: CHAR(32)
OWNER_ID: VARCHAR(1024)
DB_CODE: VARCHAR(1)
LAST_INTEREST_TIMESTAMP: DATE

MSG_CLIP_LIST

HMC_HAR_MSG_PIC: NUMBER(10)
HMC_HAR_CLIP_PIC: NUMBER(20)
BODY_SEQUENCE: NUMBER(2)
DB_CODE: VARCHAR(1)

CONSTITUENT HAR

MASTER_HAR_ID: VARCHAR(32)
HAR_ID: VARCHAR(32)

HAR

SHAZAM

DEVICE_ID: CHAR(32)
DEFAULT_PHONE_NUMBER: VARCHAR(255)
REFRESH_INTERVAL: NUMBER(5)
DB_CODE: VARCHAR(1)
DEVICE_NAME: VARCHAR(15)
SHAZAM_MODEL_ID: NUMBER(5)
ORG_ORGANIZATION_ID: CHAR(32)
SHAZAM_ACCESS_PIC: VARCHAR(255)
HAR_DEVICE_ID: CHAR(32)
DEVICE_LOCATION: VARCHAR(50)
SHAZAM_DIRECTIONAL_CODE: NUMBER(3)
REFRESH_ENABLED: NUMBER(1)
PORT_MANAGER_TIMEOUT: NUMBER(10)
UPDATED_TIMESTAMP: DATE
CREATED_TIMESTAMP: DATE
PORT_TYPE: NUMBER(1)
MESSAGE: VARCHAR(255)

SHAZAM STATUS

SHAZAM_DEVICE_ID: CHAR(32)
STATUS_CHANGE_TIME: DATE
CEN_CENTER_ID: CHAR(32)
COMM_STATUS: NUMBER(1)
BEACON_STATE: NUMBER(1)
REACON_STATE: NUMBER(1)
DEVICE_STATE_CODE: NUMBER(3)
LAST_CONTACT_TIME: DATE
LAST_ATTEMPTED_REFRESH_TIME: DATE

SHAZAM PHONE NUMBER

PORT_MANAGER_NAME: VARCHAR(255)
PHONE_NUMBER: VARCHAR(255)
PORT_MANAGER_TIMEOUT: NUMBER(10)
SHAZAM_DEVICE_ID: CHAR(32)
DB_CODE: VARCHAR(1)

HAR MSG

HAR_MSG_PIC: NUMBER(10)
USER_TRAILER: NUMBER(1)
DB_CODE: VARCHAR(1)
USER_HEADER: NUMBER(1)

STORED MESSAGE

MSG_ID: CHAR(32)
ML_ID: CHAR(32)
HMC_HAR_MSG_PIC: NUMBER(10)
CATEGORY: VARCHAR(32)
DB_CODE: VARCHAR(1)
MESSAGE_BEACON: NUMBER(1)
MSG_TYPE_CODE: NUMBER(3)
IS_MESSAGE_TEXT_MULTITEXT: NUMBER(1)
LAST_MODIFIED_BY: VARCHAR(32)
MSG_DESCRIPTION: VARCHAR(255)
MESSAGE_TEXT: VARCHAR(1024)

MESSAGE LIBRARY

ML_ID: CHAR(32)
CREATED_BY: VARCHAR(32)
DB_CODE: VARCHAR(1)
ML_NAME: VARCHAR(32)

HAR MSG_CLIP

HAR_CLIP_PIC: NUMBER(20)
CLIP_TYPE_CODE: NUMBER(3)
CLIP_POSTFIX_CODE: NUMBER(3)
RUN_LENGTH: NUMBER(5)
PRESTORED_SLOT_NUMBER: NUMBER(3)
RUN_LENGTH_CODE: NUMBER(3)
HCA_HAR_CLIP_AUDIO_ID: CHAR(32)
DB_CODE: VARCHAR(1)
CLIP_DESCRIPTION: VARCHAR(255)
CLIP_TEXT: CLOB
AUDIO_CLIP_MANAGER_ID: VARCHAR(1024)

DMS

DEVICE_ID: CHAR(32)
DMS_MODEL_ID: NUMBER(5)
ORG_ORGANIZATION_ID: CHAR(32)
DATA_BITS: NUMBER(1)
BAUD_RATE: NUMBER(5)
STOP_BITS: NUMBER(1)
PORT_TYPE: NUMBER(1)
POLLING_ENABLED: NUMBER(1)
PARTY: NUMBER(1)
DEFAULT_PAGE_OFF_TIME: NUMBER(3)
DEFAULT_PAGE_ON_TIME: NUMBER(3)
DROP_ADDRESS: NUMBER(5)
INITIAL_RESPONSE_TIMEOUT: NUMBER(10)
FLOW_CONTROL: NUMBER(1)
ENABLE_DEVICE_LOG: NUMBER(1)
PORT_MANAGER_TIMEOUT: NUMBER(10)
VMS_CHARACTER_HEIGHT_PIXELS: NUMBER(3)
DB_CODE: VARCHAR(1)
HAR_DEVICE_ID: CHAR(32)
DEFAULT_PHONE_NUMBER: VARCHAR(255)
COMM_LOSS_TIMEOUT: NUMBER(10)
DEFAULT_JUSTIFICATION_LINE: NUMBER(3)
BEACON_TYPE: NUMBER(5)
POLL_INTERVAL: NUMBER(5)
SIGN_TYPE: NUMBER(3)
CREATED_TIMESTAMP: DATE
VMS_MAX_PAGES: NUMBER(3)
VMS_SIGN_HEIGHT_PIXELS: NUMBER(5)
VMS_SIGN_WIDTH_PIXELS: NUMBER(3)
SHAZAM_BEACON_STATE: NUMBER(1)
UPDATED_TIMESTAMP: DATE
VMS_CHARACTER_WIDTH_PIXELS: NUMBER(3)
SHAZAM_IS_MESSAGE_TEXT_MULTITEXT: NUMBER(1)
DMS_SHAZAM_MSG: VARCHAR(1024)
COMMUNITY_STRING: VARCHAR(10)
CEN_ALERT_CENTER_ID: VARCHAR(32)
TRAVEL_TIME_QUEUE_LEVEL: NUMBER(3)
TOLL_RATE_QUEUE_LEVEL: NUMBER(5)
USE_ONWRITE_SCHEDULE_INDICATOR: NUMBER(2)
ENABLED_SPECIFIC_TIME_INDICATOR: NUMBER(2)
EVENT_INT_EXT_SYSTEM: VARCHAR(255)
EVENT_INT_EXT_AGENCY: VARCHAR(255)
EXTERNAL_EVENT_SNO: NUMBER(1)
TCP_HOST: VARCHAR(16)
TCP_PORT: NUMBER(5)
DEFAULT_FONT_NUMBER: NUMBER(1)
DEFAULT_LINE_SPACING: NUMBER(1)

DMS STATUS

DMS_DEVICE_ID: CHAR(32)
LAST_ATTEMPTED_POLL_TIME: DATE
SHORT_ERROR_STATUS: NUMBER(5)
CEN_CENTER_ID: CHAR(32)
DEVICE_STATE_CODE: NUMBER(3)
PIXEL_TEST: NUMBER(1)
COMM_STATUS: NUMBER(1)
BEACON_STATE: NUMBER(1)
STATUS_LOG_DATE: DATE
LAST_CONTACT_TIME: DATE
DMS_INITIALIZED: NUMBER(1)
STATUS_CHANGE_TIME: DATE
CURRENT_MESSAGE_TEXT: VARCHAR(1024)

DMS PHONE NUMBER

PORT_MANAGER_NAME: VARCHAR(255)
PHONE_NUMBER: VARCHAR(255)
PORT_MANAGER_TIMEOUT: NUMBER(10)
DMS_DEVICE_ID: CHAR(32)
DB_CODE: VARCHAR(1)

DMS FONT

DMS_DEVICE_ID: CHAR(32)
FONT_FONT_ID: NUMBER(3)
DB_CODE: VARCHAR(1)

FONT

FONT_ID: NUMBER(3)
FONT_HEIGHT: NUMBER(3)
DB_CODE: VARCHAR(1)
FONT_WIDTH: NUMBER(3)

DMS

TSS

DEVICE_ID: VARCHAR(32)
PORT_TYPE: NUMBER(1)
DATA_BITS: NUMBER(1)
POLL_INTERVAL_SECONDS: NUMBER(5)
DEVICE_NAME: VARCHAR(15)
DEVICE_LOCATION: VARCHAR(50)
TSS_MODEL_ID: NUMBER(5)
ORG_ORGANIZATION_ID: CHAR(32)
FLOW_CONTROL: NUMBER(1)
PORT_MANAGER_TIMEOUT: NUMBER(10)
DROP_ADDRESS: NUMBER(5)
DB_CODE: VARCHAR(1)
DEFAULT_PHONE_NUMBER: VARCHAR(255)
INITIAL_RESPONSE_TIMEOUT: NUMBER(10)
BAUD_RATE: NUMBER(5)
STOP_BITS: NUMBER(1)
CREATED_TIMESTAMP: DATE
ENABLE_DEVICE_LOG: NUMBER(1)
PARTY: NUMBER(1)
UPDATED_TIMESTAMP: DATE
CEN_ALERT_CENTER_ID: VARCHAR(32)

TSS STATUS

TSS_DEVICE_ID: VARCHAR(32)
COMM_STATUS: NUMBER(1)
LAST_CONTACT_TIME: DATE
DEVICE_STATE_CODE: NUMBER(3)

TSS PHONE NUMBER

PORT_MANAGER_NAME: VARCHAR(255)
PHONE_NUMBER: VARCHAR(255)
PORT_MANAGER_TIMEOUT: NUMBER(10)
TSS_DEVICE_ID: VARCHAR(32)
DB_CODE: VARCHAR(1)

TSS_ZONE GROUP

TSS_DEVICE_ID: VARCHAR(32)
GROUP_NUMBER: NUMBER(10)
DEFAULT_SPEED: NUMBER(3)
DIRECTION_CODE: NUMBER(3)
DESCRIPTION: VARCHAR(255)
DB_CODE: VARCHAR(1)

TSS_ZONE

TSS23_GROUP_NUMBER: NUMBER(10)
ZONE_NUMBER: NUMBER(3)
TSS23_TSS_DEVICE_ID: VARCHAR(32)
DB_CODE: VARCHAR(1)

DEVICE EVENT

EVENT_EVENT_ID: CHAR(32)
RPL_ID: CHAR(32)
DEVICE_USAGE_CODE: NUMBER(3)
DEVICE_ID: CHAR(32)

RESPONSE PLAN ITEM

RPL_ID: CHAR(32)
IS_MESSAGE_TEXT_MULTITEXT: NUMBER(1)
DEVICE_CODE: NUMBER(3)
DB_CODE: VARCHAR(1)
HMC_HAR_MSG_PIC: NUMBER(10)
RPL_BEACON_STATE: NUMBER(1)
EVENT_EVENT_ID: CHAR(32)
TARGET_ID: CHAR(32)
RPL_MSG_DESCRIPTION: VARCHAR(255)
RPL_MESSAGE_TEXT: VARCHAR(1024)
TARGET_ID: VARCHAR(1024)

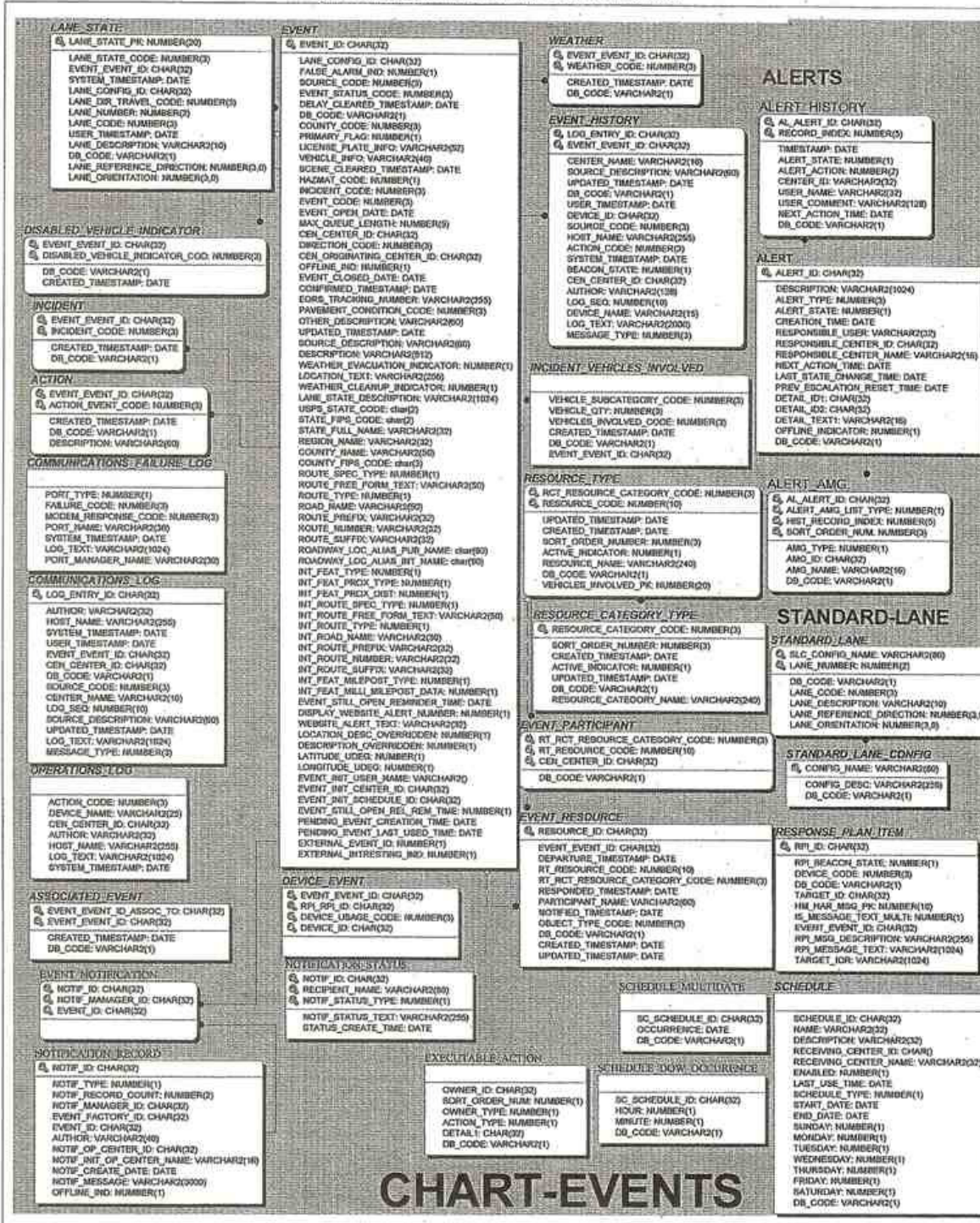
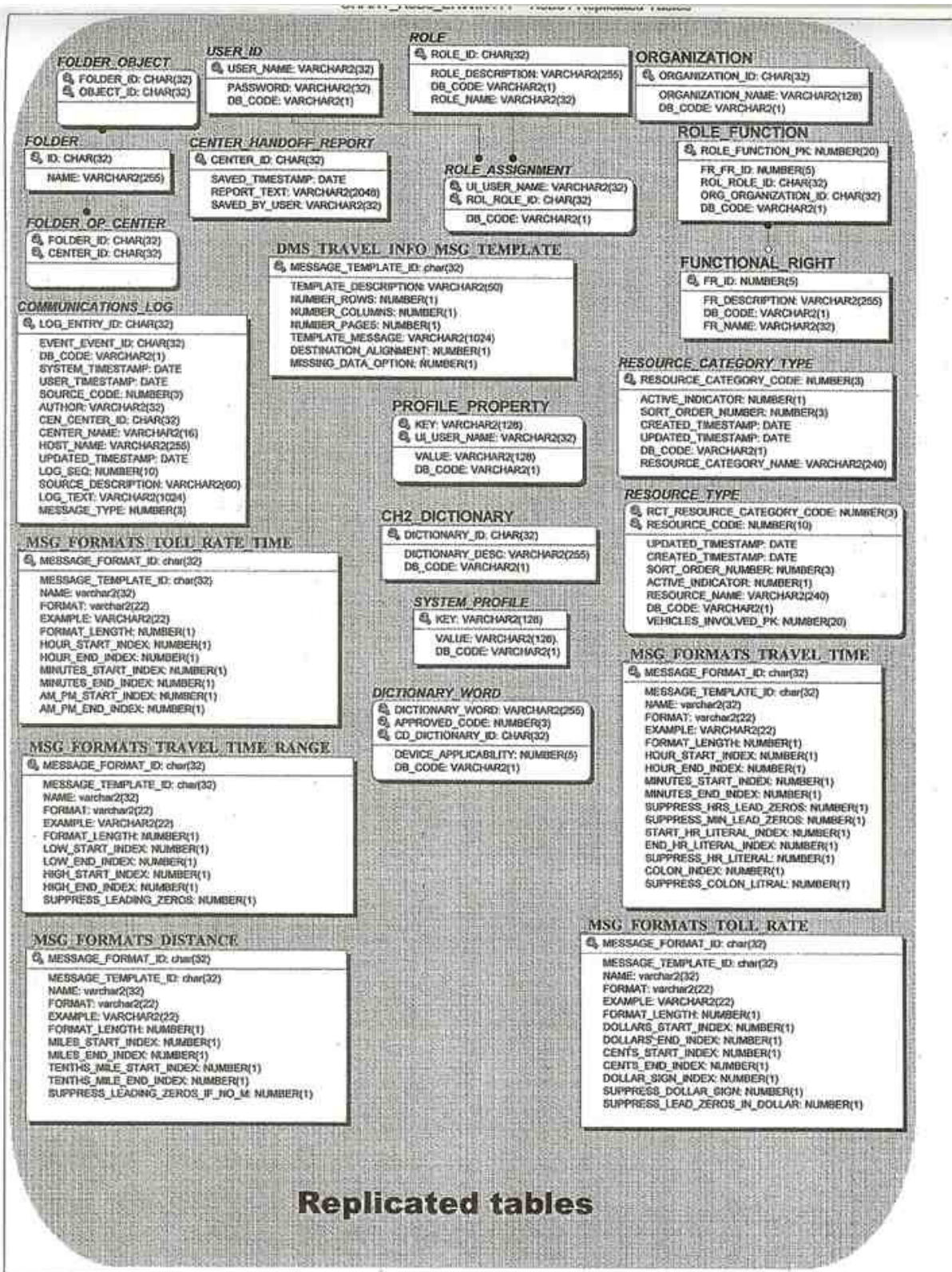
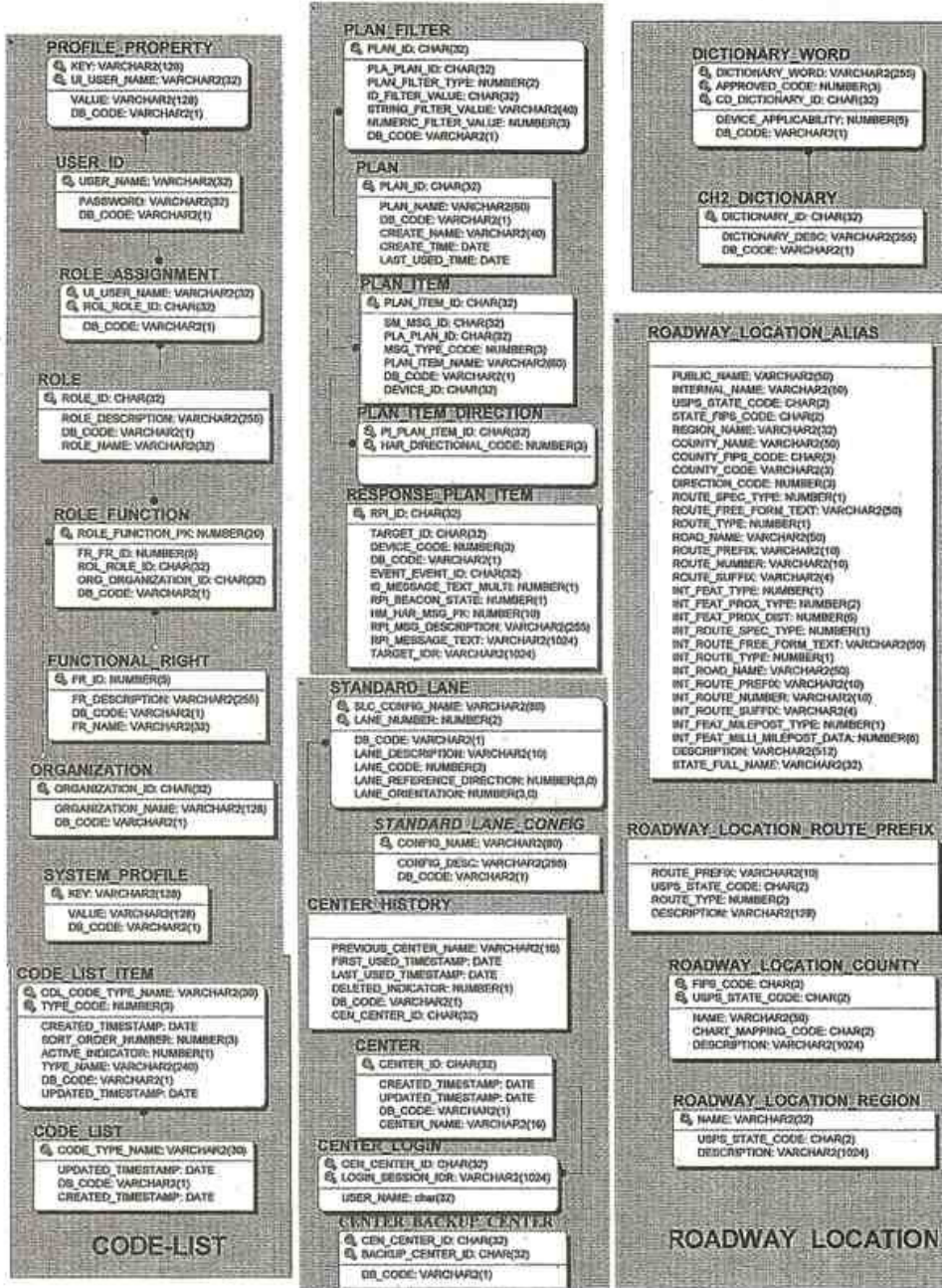


CHART-EVENTS

2.1 / 2.2 -- 11:46:44 AM, 12/17/2008



1, 1 / 1, 1 - 11:47:38 AM, 12/17/2008



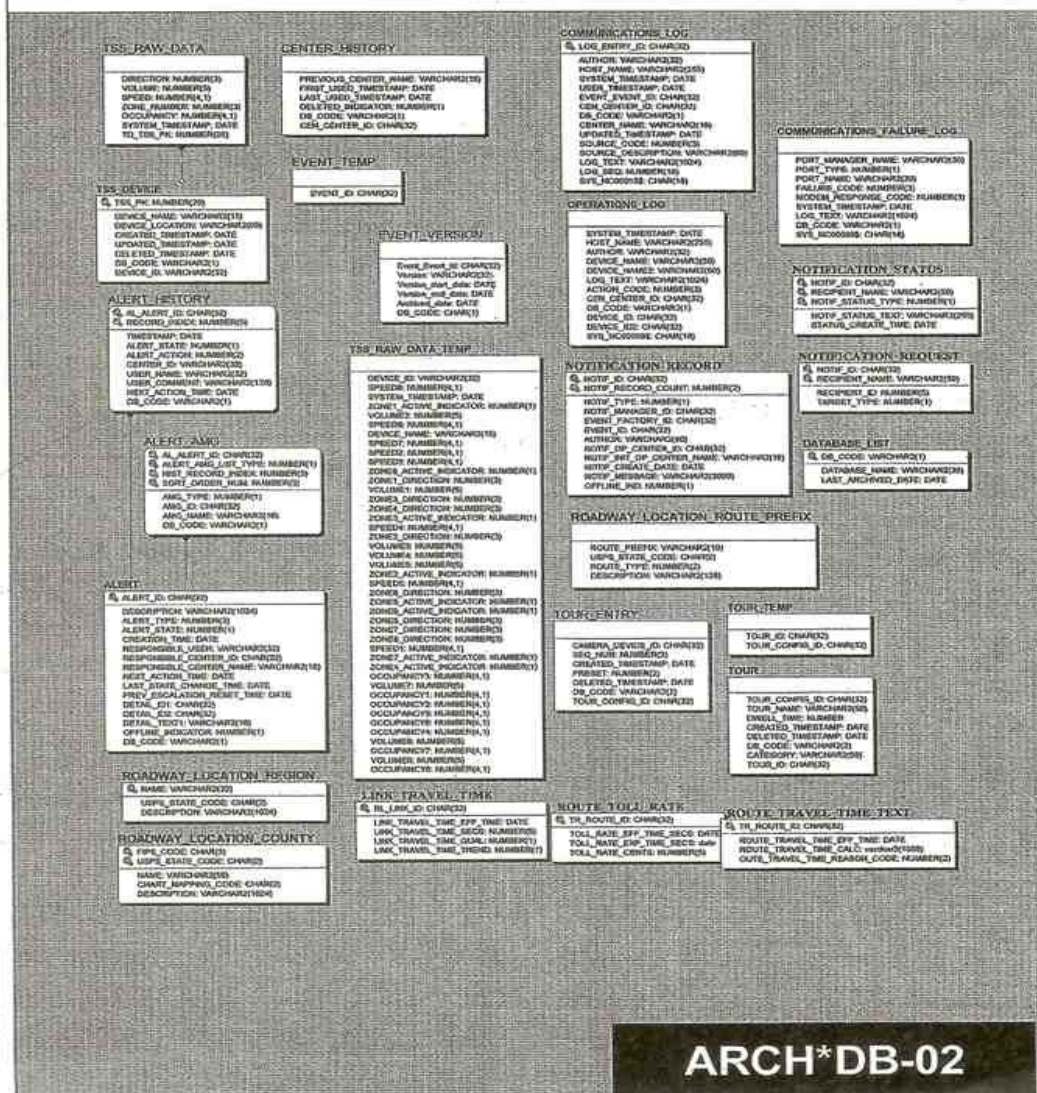


Figure 2-6 ERD for CHART R3B3

2.4.1.1.2.2 Function to Entity Matrix Report

- Create, Retrieve, Update, Delete (CRUD) matrix cross-references business functions to entities and shows the use of the entities by those functions. This report will be generated as part of the CHART O&M Guide.

2.4.1.1.2.3 Table Definition Report –

2.4.1.1.2.3.1 New Tables for TravelRouteModule in CHART R3B3 Live database

ROADWAY_LINK

LINK_ID	char(32)	NOT NULL
EXT_SYS_NAME	varchar2(10)	NOT NULL
EXT_LINK_ID	varchar2(9)	NOT NULL
LINK_NAME	varchar2(50)	
USPS_STATE_CODE	char(2)	
STATE_FIPS_CODE	char(2)	
COUNTY_NAME	varchar2(50)	
COUNTY_FIPS_CODE	varchar2(3)	
ROUTE_SPEC_TYPE	number(1)	
ROUTE_FREE_FORM_TEXT	varchar2(50)	
ROUTE_TYPE	number(1)	
ROUTE_PREFIX	varchar2(10)	
ROUTE_NUMBER	varchar2(10)	
ROUTE_SUFFIX	varchar2(10)	
MILLI_MILES	number(5)	
START_LAT_UDEG	number(9)	
START_LONG_UDEG	number(10)	
END_LAT_UDEG	number(9)	
END_LONG_UDEG	number(10)	

LINK_TRAVEL_TIME

RL_LINK_ID	char(32)	NOT NULL
LINK_TRAVEL_TIME_EFF_TIME	date	NOT NULL
LINK_TRAVEL_TIME_SECS	number(5)	NOT NULL
LINK_TRAVEL_TIME_QUAL	number(1)	NOT NULL
LINK_TRAVEL_TIME_TREND	number(1)	NOT NULL

TRAVEL_ROUTE

TRAVEL_ROUTE_ID	char(32)	NOT NULL
NAME	varchar2(50)	NOT NULL
MILLI_MILES	number(6)	
USER_LOCATION_INDICATOR	number(1)	NOT NULL
PRIMARY_DEST_TEXT	varchar2(30)	
TRAVEL_TIME_ENABLED_INDICATOR	number(1)	NOT NULL
MIN_TRAVEL_TIME_MINS	number(3)	
MAX_TRAVEL_TIME_MINS	number(3)	
ALERT_TRAVEL_TIME_MINS	number(3)	
ALERTS_ENABLED_INDICATOR	number(1)	NOT NULL

ALERT_OP_CENTER	char(32)	
NOTIFS_ENABLED_INDICATOR	number(1)	NOT NULL
NOTIF_GROUP	varchar2(33)	
TOLL_RATE_EXT_SYS_NAME	varchar2(35)	
TOLL_RATE_EXT_START_ID	varchar2(10)	
TOLL_RATE_EXT_END_ID	varchar2(10)	
TOLL_RATE_EXT_DESC	varchar2(25)	
TOLL_RATE_ENABLED_INDICATOR	number(1)	NOT NULL

TRAVEL_ROUTE_STATE

TR_ID	char(32)	NOT NULL
SORT_ORDER_NUMBER	number(2)	NOT NULL
USPS_STATE_CODE	char(2)	
STATE_FIPS_CODE	char(2)	

TRAVEL_ROUTE_COUNTY

TR_ID	char(32)	NOT NULL
SORT_ORDER_NUMBER	number(2)	NOT NULL
COUNTY_NAME	varchar2(50)	
COUNTY_FIPS_CODE	char(3)	

TRAVEL_ROUTE_ROUTE_SPEC

TR_ID	char(32)	not null
SORT_ORDER_NUMBER	number(2)	NOT NULL
ROUTE_SPEC_TYPE	number(1)	
ROUTE_FREE_FORM_TEXT	varchar2(50)	
ROUTE_TYPE	number(1)	
ROUTE_PREFIX	varchar2(10)	
ROUTE_NUMBER	varchar2(10)	
ROUTE_SUFFIX	varchar2(10)	

TRAVEL_ROUTE_DEST

TR_ID	char(32)	not null
SORT_ORDER_NUMBER	number(1)	NOT NULL
ALT_DEST_TEXT	varchar2(30)	

TRAVEL_ROUTE_CONSUMER

TR_ID	char(32)	not null
SORT_ORDER_NUMBER	number(2)	NOT NULL
CONSUMER_ID	char(32)	

TRAVEL_ROUTE_LINK

TR_ID	char(32)	not null
SORT_ORDER_NUMBER	number(4)	NOT NULL
RL_ID	char(32)	NOT NULL

PERCENT	number (3)	NOT NULL
MIN_ALLOWED_QUALITY	number (3)	NOT NULL

ROUTE_TOLL_RATE

TR_ID	char (32)	NOT NULL
TOLL_RATE_EFF_TIME	date	NOT NULL
TOLL_RATE_EXP_TIME	date	
TOLL_RATE_CENTS	number (5)	NOT NULL

ROUTE_TRAVEL_TIME

TR_ID	CHAR (32)	NOT NULL
ROUTE_TRAVEL_TIME_EFF_TIME	DATE	NOT NULL
ROUTE_TRAVEL_TIME_SECS	NUMBER (5)	NOT NULL
ROUTE_TRAVEL_TIME_QUAL	NUMBER (1)	NOT NULL
ROUTE_TRAVEL_TIME_TREND	NUMBER (1)	NOT NULL

ROUTE_TRAVEL_TIME_TEXT

TR_ID	CHAR (32)	NOT NULL
ROUTE_TRAVEL_TIME_EFF_TIME	DATE	NOT NULL
ROUTE_TRAVEL_TIME_CALC	VARCHAR2 (1000)	NOT NULL
ROUTE_TRAVEL_TIME_REASON_CODE	NUMBER (2)	NOT NULL

RAW_LINK_DATA

SYSTEM_TIMESTAMP	DATE	NOT NULL
EXT_SYS_NAME	VARCHAR2 (35)	NOT NULL
EXT_LINK_ID	CHAR (8)	NOT NULL
LINK_TRAVEL_TIME_EFF_TIME	DATE	NOT NULL
LINK_TRAVEL_TIME_SECS	NUMBER (5)	NOT NULL
LINK_TRAVEL_TIME_QUAL	NUMBER (1)	NOT NULL

RAW_TOLL_DATA

SYSTEM_TIMESTAMP	DATE	NOT NULL
EXT_SYS_NAME	VARCHAR2 (35)	NOT NULL
EXT_SYS_START_ID	CHAR (32)	NOT NULL
EXT_SYS_END_ID	CHAR (32)	NOT NULL
TOLL_RATE_EFF_TIME	DATE	NOT NULL
TOLL_RATE_EXP_TIME	DATE	
TOLL_RATE_CENTS	NUMBER (5)	NOT NULL

2.4.1.1.2.3.2 New Tables for External DMS in CHART R3B3 Live database

Device_location

DEVICE_ID	VARCHAR2 (32)	NOT NULL
LOCATION_TEXT	VARCHAR2 (1024)	
LOCATION_DESC_OVERRIDDEN	NUMBER (1)	

COUNTY_NAME	VARCHAR2 (50)
COUNTY_FIPS_CODE	CHAR (3)
COUNTY_CODE	NUMBER (3)
USPS_STATE_CODE	CHAR (2)
STATE_FULL_NAME	VARCHAR2 (32)
STATE_FIPS_CODE	CHAR (2)
REGION_NAME	VARCHAR2 (32)
ROUTE_SPEC_TYPE	NUMBER (1)
ROUTE_FREE_FORM_TEXT	VARCHAR2 (40)
ROUTE_TYPE	NUMBER (1)
ROUTE_PREFIX	VARCHAR2 (10)
ROUTE_NUMBER	VARCHAR2 (10)
ROUTE_SUFFIX	VARCHAR2 (10)
INT_FEAT_TYPE	NUMBER (1)
INT_FEAT_PROX_TYPE	NUMBER (2)
INT_FEAT_PROX_DIST	NUMBER (6)
ROAD_NAME	VARCHAR2 (50)
INT_ROUTE_SPEC_TYPE	NUMBER (1)
INT_ROUTE_FREE_FORM_TEXT	VARCHAR2 (40)
INT_ROUTE_TYPE	NUMBER (1)
INT_ROAD_NAME	VARCHAR2 (50)
INT_ROUTE_PREFIX	VARCHAR2 (10)
INT_ROUTE_NUMBER	VARCHAR2 (10)
INT_ROUTE_SUFFIX	VARCHAR2 (4)
INT_FEAT_MILEPOST_TYPE	NUMBER (1)
INT_FEAT_MILLI_MILEPOST_DATA	NUMBER (6)
ROADWAY_LOC_ALIAS_PUB_NAME	VARCHAR2 (32)
ROADWAY_LOC_ALIAS_INT_NAME	VARCHAR2 (32)
LATITUDE_UDEG	NUMBER (8)
LONGITUDE_UDEG	NUMBER (9)
GEOLOC_SOURCE_TYPE	NUMBER (2)
GEOLOC_SOURCE_DESC	VARCHAR2 (35)
SHOW_ROUTE_NAME	NUMBER (1)
SHOW_INT_ROUTE_NAME	NUMBER (1)
DIRECTION_CODE	NUMBER (3)
DEVICE_TYPE	VARCHAR2 (10)

DMS_TRAV_TIME_SCHEDULE

DMS_DEVICE_ID	CHAR (32)	NOT NULL
START_HOUR	NUMBER (2)	NOT NULL
START_MIN	NUMBER (2)	NOT NULL
END_HOUR	NUMBER (2)	NOT NULL
END_MIN	NUMBER (2)	NOT NULL

DMS_TRAV_ROUTE_MSG

DMS_DEVICE_ID	CHAR (32)	NOT NULL
MSG_ID	CHAR (32)	NOT NULL
TEMPLATE_ID	CHAR (32)	NOT NULL
AUTO_ROW_POSITIONING_INDICATOR	NUMBER (1)	NOT NULL

DMS_TRAV_ROUTE_MSG_ROUTE

DTRM_MSG_ID	CHAR(32)	NOT NULL
TRAVEL_ROUTE_ID	CHAR(32)	NOT NULL

DMS_RELATED_ROUTE

DMS_DEVICE_ID	CHAR(32)	NOT NULL
TRAVEL_ROUTE_ID	CHAR(32)	NOT NULL

GEO_AREA

GEO_AREA_ID	CHAR(32)	NOT NULL
NAME	VARCHAR2(15)	NOT NULL
DESCRIPTION	VARCHAR2(60)	NOT NULL

GEO_AREA_POINT

GA_GEO_AREA_ID	CHAR(32)	NOT NULL
SORT_ORDER_NUMBER	NUMBER(5)	NOT NULL
LATITUDE_UDEG	NUMBER(8)	NOT NULL
LONGITUDE_UDEG	NUMBER(9)	NOT NULL

EXTERNAL_OBJECT_EXCLUSION

EXCLUSION_ID	CHAR(32)	NOT NULL
EXTERNAL_OBJECT_ID	VARCHAR2(35)	NOT NULL
EXTERNAL_OBJECT_TYPE	NUMBER(3)	NOT NULL
EXTERNAL_SYSTEM	VARCHAR2(35)	NOT NULL
EXTERNAL_AGENCY	VARCHAR2(35)	NOT NULL

EXTERNAL_EVENT_FILTER

RULE_ID	CHAR(32)	NOT NULL
RULE	VARCHAR2(2048)	NOT NULL

2.4.1.1.2.3.3 Tables Modified for External DMS in CHART R3B3 Live database

DMS table

Existing / Deleted (“(-)”) columns

DEVICE_ID	NOT NULL	CHAR(32)
DMS_MODEL_ID	NOT NULL	NUMBER(5)
ORG_ORGANIZATION_ID	NOT NULL	CHAR(32)
DB_CODE		VARCHAR2(1)
DEVICE_NAME	NOT NULL	VARCHAR2(15)

(-) DEVICE_LOCATION	NOT NULL	VARCHAR2 (60)
HAR_DEVICE_ID		CHAR (32)
COMM_LOSS_TIMEOUT	NOT NULL	NUMBER (10)
DEFAULT_JUSTIFICATION_LINE	NOT NULL	NUMBER (3)
DEFAULT_PAGE_OFF_TIME	NOT NULL	NUMBER (3)
DEFAULT_PAGE_ON_TIME	NOT NULL	NUMBER (3)
DROP_ADDRESS	NOT NULL	NUMBER (5)
INITIAL_RESPONSE_TIMEOUT	NOT NULL	NUMBER (10)
BEACON_TYPE	NOT NULL	NUMBER (3)
SIGN_TYPE	NOT NULL	NUMBER (3)
DEFAULT_PHONE_NUMBER		VARCHAR2 (25)
(-) DMS_DIRECTIONAL_CODE		NUMBER (3)
POLL_INTERVAL	NOT NULL	NUMBER (5)
POLLING_ENABLED	NOT NULL	NUMBER (1)
PORT_TYPE	NOT NULL	NUMBER (1)
PORT_MANAGER_TIMEOUT	NOT NULL	NUMBER (10)
BAUD_RATE	NOT NULL	NUMBER (6)
DATA_BITS	NOT NULL	NUMBER (1)
FLOW_CONTROL	NOT NULL	NUMBER (1)
PARITY	NOT NULL	NUMBER (1)
STOP_BITS	NOT NULL	NUMBER (1)
ENABLE_DEVICE_LOG	NOT NULL	NUMBER (1)
VMS_CHARACTER_HEIGHT_PIXELS	NOT NULL	NUMBER (3)
VMS_CHARACTER_WIDTH_PIXELS	NOT NULL	NUMBER (3)
VMS_MAX_PAGES	NOT NULL	NUMBER (3)
VMS_SIGN_HEIGHT_PIXELS	NOT NULL	NUMBER (5)
VMS_SIGN_WIDTH_PIXELS	NOT NULL	NUMBER (5)
CREATED_TIMESTAMP		DATE
UPDATED_TIMESTAMP		DATE
SHAZAM_BEACON_STATE	NOT NULL	NUMBER (1)
SHAZAM_IS_MESSAGE_TEXT_MULTI	NOT NULL	NUMBER (1)
DMS_SHAZAM_MSG		
VARCHAR2 (1024)		
COMMUNITY_STRING		VARCHAR2 (16)
CEN_ALERT_CENTER_ID		CHAR (32)

New (“+”) columns

+TRAVEL_TIME_QUEUE_LEVEL	NUMBER (5)	NOT NULL
+TOLL_RATE_QUEUE_LEVEL	NUMBER (5)	NOT NULL
+OVERRIDE_SCHEDULE_INDICATOR	NUMBER (2)	NOT NULL
+ENABLED_SPECIFIC_TIMES_INDICATOR	NUMBER (2)	NOT NULL
+EXTERNAL_SYSTEM	VARCHAR2 (35)	
+EXTERNAL_AGENCY	VARCHAR2 (35)	
+EXTERNAL_OBJECT_ID	VARCHAR2 (35)	
+EXTERNAL_OBJECT_INDICATOR	NUMBER (1)	NOT NULL
+TCP_HOST	VARCHAR2 (30)	
+TCP_PORT	NUMBER (5)	
+DEFAULT_FONT_NUMBER	NUMBER (2)	
+DEFAULT_LINE_SPACING	NUMBER (1)	

TSS

Existing / Deleted (“(-)”) columns

DEVICE_ID	NOT NULL VARCHAR2 (32)
TSS_MODEL_ID	NOT NULL NUMBER (5)
ORG_ORGANIZATION_ID	NOT NULL CHAR (32)
DB_CODE	VARCHAR2 (1)
DEVICE_NAME	NOT NULL VARCHAR2 (15)
(-) DEVICE_LOCATION	NOT NULL
VARCHAR2 (60)	
DROP_ADDRESS	NOT NULL NUMBER (5)
INITIAL_RESPONSE_TIMEOUT	NOT NULL NUMBER (10)
DEFAULT_PHONE_NUMBER	VARCHAR2 (25)
POLL_INTERVAL_SECS	NOT NULL NUMBER (5)
PORT_TYPE	NOT NULL NUMBER (1)
PORT_MANAGER_TIMEOUT	NOT NULL NUMBER (10)
BAUD_RATE	NOT NULL NUMBER (6)
DATA_BITS	NOT NULL NUMBER (1)
FLOW_CONTROL	NOT NULL NUMBER (1)
PARITY	NOT NULL NUMBER (1)
STOP_BITS	NOT NULL NUMBER (1)
ENABLE_DEVICE_LOG	NOT NULL NUMBER (1)
CREATED_TIMESTAMP	DATE
UPDATED_TIMESTAMP	DATE
CEN_ALERT_CENTER_ID	CHAR (32)

New (“+”) columns

+EXTERNAL_SYSTEM	VARCHAR2 (35)
+EXTERNAL_AGENCY	VARCHAR2 (35)
+EXTERNAL_OBJECT_ID	VARCHAR2 (35)
+EXTERNAL_OBJECT_INDICATOR	NUMBER (1) NOT NULL
+TCP_HOST	VARCHAR2 (30)
+TCP_PORT	NUMBER (5)

SHAZAM

Existing / Deleted (“(-)”) columns

DEVICE_ID	NOT NULL CHAR (32)
SHAZAM_MODEL_ID	NOT NULL NUMBER (5)
ORG_ORGANIZATION_ID	NOT NULL CHAR (32)
DB_CODE	VARCHAR2 (1)
DEVICE_NAME	NOT NULL VARCHAR2 (15)
(-) DEVICE_LOCATION	NOT NULL
VARCHAR2 (60)	
HAR_DEVICE_ID	CHAR (32)
SHAZAM_ACCESS_PIN	VARCHAR2 (3)
DEFAULT_PHONE_NUMBER	NOT NULL VARCHAR2 (25)
(-) SHAZAM_DIRECTIONAL_CODE	
NUMBER (3)	
REFRESH_INTERVAL	NUMBER (5)

REFRESH_ENABLED	NOT NULL	NUMBER(1)
PORT_TYPE	NOT NULL	NUMBER(1)
PORT_MANAGER_TIMEOUT	NOT NULL	NUMBER(10)
CREATED_TIMESTAMP		DATE
UPDATED_TIMESTAMP		DATE
MESSAGE		
VARCHAR2(256)		

HAR

Existing / Deleted (“(-)”) columns

DEVICE_ID	NOT NULL	CHAR(32)
HAR_MODEL_ID	NOT NULL	NUMBER(5)
ORG_ORGANIZATION_ID	NOT NULL	CHAR(32)
DB_CODE		VARCHAR2(1)
DEVICE_NAME	NOT NULL	VARCHAR2(15)
(-) DEVICE_LOCATION		NOT NULL
VARCHAR2(60)		
HAR_ACCESS_PIN	NOT NULL	VARCHAR2(7)
DEFAULT_PHONE_NUMBER	NOT NULL	VARCHAR2(25)
DEFAULT_MONITOR_PHONE_NUMBER		VARCHAR2(25)
MAX_TIME		NUMBER(5)
PORT_TYPE	NOT NULL	NUMBER(1)
PORT_MANAGER_TIMEOUT	NOT NULL	NUMBER(10)
MONITOR_PORT_TYPE		NUMBER(1)
MONITOR_PORT_MANAGER_TIMEOUT		NUMBER(10)
DEFAULT_HEADER_CLIP_PK	NOT NULL	NUMBER(20)
DEFAULT_BODY_CLIP_PK	NOT NULL	NUMBER(20)
DEFAULT_TRAILER_CLIP_PK	NOT NULL	NUMBER(20)
CREATED_TIMESTAMP		DATE
UPDATED_TIMESTAMP		DATE
ENABLE_DEVICE_LOG	NOT NULL	NUMBER(1)
MASTER_HAR_ID	NOT NULL	CHAR(32)

CAMERA

Existing / Deleted (“(-)”) columns

DEVICE_ID	NOT NULL	CHAR(32)
CAMERA_MODEL_ID	NOT NULL	NUMBER(3)
ORG_ORGANIZATION_ID	NOT NULL	CHAR(32)
DEVICE_NAME	NOT NULL	VARCHAR2(50)
LOCATION_PROFILE_TYPE		NUMBER(3)
LOCATION_PROFILE_ID		CHAR(32)
TMDD_CCTV_IMAGE		NUMBER(2)
CAMERA_NUMBER		NUMBER(5)
CAMERA_CONTROLLABLE	NOT NULL	NUMBER(1)
TMDD_CONTROL_TYPE		NUMBER(2)
TMDD_REQUEST_COMMAND_TYPES	NOT NULL	NUMBER(5)
ENABLE_DEVICE_LOG	NOT NULL	NUMBER(1)
VIDEO_CONNECTION_ID	NOT NULL	CHAR(32)

VIDEO_CONNECTION_TYPE	NOT NULL	NUMBER(2)
NO_VIDEO_AVAIL_INDICATOR	NOT NULL	NUMBER(1)
(-) DEVICE_LOCATION_DESC		VARCHAR2(50)
TMDD_DEVICE_NAME		VARCHAR2(50)
POLL_INTERVAL_CONTROLLED_SECS		NUMBER(5)
POLLING_ENABLED_UNCONTROLLED		NUMBER(1)
DEFAULT_CAMERA_TITLE		VARCHAR2(24)
DEFAULT_CAMERA_TITLE_LINE2		VARCHAR2(24)
CONTROL_CONNECTION_TYPE		NUMBER(1)
CONTROL_CONNECTION_ID		CHAR(32)
POLL_INTERVAL_UNCTRLD_SECS		NUMBER(4)
DB_CODE		VARCHAR2(1)
CREATED_TIMESTAMP		DATE
UPDATED_TIMESTAMP		DATE
DSP_STATUS_ENABLED		NUMBER(1)
DSP_STATUS_LENGTH		NUMBER(5)

2.4.1.1.2.3.4 New Tables for Travel Route Message Templates

DMS_TRAVEL_ROUTE_MSG_TEMPLATE (Replicated)

MESSAGE_TEMPLATE_ID	CHAR(32)
TEMPLATE_DESCRIPTION	VARCHAR2(50)
NUMBER_ROWS	NUMBER(1)
NUMBER_COLUMNS	NUMBER(1)
NUMBER_PAGES	NUMBER(1)
TEMPLATE_MESSAGE	VARCHAR2(1024)
DESTINATION_ALIGNMENT	NUMBER(1)
MISSING_DATA_OPTION	NUMBER(1)

MSG_FORMATS_TOLL_RATE_TIME (Replicated)

MESSAGE_FORMAT_ID (KEY)	CHAR(32)	
MESSAGE_TEMPLATE_ID	CHAR(32)	NULLABLE
NAME	VARCHAR(50)	
FORMAT	VARCHAR2(22)	
EXAMPLE	VARCHAR2(22)	
FORMAT_LENGTH	NUMBER(2)	
HOURLY_START_INDEX	NUMBER(2)	
HOURLY_END_INDEX	NUMBER(2)	
MINUTES_START_INDEX	NUMBER(2)	
MINUTES_END_INDEX	NUMBER(2)	
AM_PM_START_INDEX	NUMBER(2)	
AM_PM_END_INDEX	NUMBER(2)	

MSG_FORMATS_TRAVEL_TIME (Replicated)

MESSAGE_FORMAT_ID (KEY)	CHAR(32)	
MESSAGE_TEMPLATE_ID	CHAR(32)	NULLABLE
NAME	VARCHAR(50)	
FORMAT	VARCHAR2(22)	
EXAMPLE	VARCHAR2(22)	
FORMAT_LENGTH	NUMBER(2)	

HOURL_START_INDEX	NUMBER (2)
HOURL_END_INDEX	NUMBER (2)
SUPPRESS_HRS_LEAD_ZEROS	NUMBER (1)
MINUTES_START_INDEX	NUMBER (2)
MINUTES_END_INDEX	NUMBER (2)
SUPPRESS_MIN_LEAD_ZEROS	NUMBER (1)
START_HR_LITERAL_INDEX	NUMBER (2)
END_HR_LITERAL_INDEX	NUMBER (2)
SUPPRESS_HR_LITERAL	NUMBER (1)
COLON_INDEX	NUMBER (2)
SUPPRESS_COLON_LITERAL	NUMBER (1)

MSG_FORMATS_TRAVEL_TIME_RANGE (Replicated)

MESSAGE_FORMAT_ID (KEY)	CHAR (32)	
MESSAGE_TEMPLATE_ID	CHAR (32)	NULLABLE
NAME	VARCHAR2 (50)	
FORMAT	VARCHAR2 (22)	
EXAMPLE	VARCHAR2 (22)	
FORMAT_LENGTH	NUMBER (2)	
LOW_START_INDEX	NUMBER (2)	
LOW_END_INDEX	NUMBER (2)	
HIGH_START_INDEX	NUMBER (2)	
HIGH_END_INDEX	NUMBER (2)	
SUPPRESS_LEADING_ZEROS	NUMBER (1)	

MSG_FORMATS_TOLL_RATE (Replicated)

MESSAGE_FORMAT_ID (KEY)	CHAR (32)	
MESSAGE_TEMPLATE_ID	CHAR (32)	NULLABLE
NAME	VARCHAR2 (50)	
FORMAT	VARCHAR2 (22)	
EXAMPLE	VARCHAR2 (22)	
FORMAT_LENGTH	NUMBER (2)	
DOLLARS_START_INDEX	NUMBER (2)	
DOLLARS _END_INDEX	NUMBER (2)	
CENTS_START_INDEX	NUMBER (2)	
CENTS _END_INDEX	NUMBER (2)	
DOLLAR_SIGN_INDEX	NUMBER (2)	
SUPPRESS_DOLLAR_SIGN	NUMBER (1)	
SUPPRESS_LEAD_ZEROS_IN_DOLLAR	NUMBER (1)	

MSG_FORMATS_DISTANCE (Replicated)

MESSAGE_FORMAT_ID (KEY)	CHAR (32)	
MESSAGE_TEMPLATE_ID	CHAR (32)	NULLABLE
NAME	VARCHAR2 (50)	
FORMAT	VARCHAR2 (22)	
EXAMPLE	VARCHAR2 (22)	
FORMAT_LENGTH	NUMBER (2)	
MILES_START_INDEX	NUMBER (2)	
MILES_END_INDEX	NUMBER (2)	
TENTHS_MILE_START_INDEX	NUMBER (2)	
TENTHS _MILE_END_INDEX	NUMBER (2)	
SUPPRESS_LEAD_ZEROS_IF_NO_MILES	NUMBER (1)	

2.4.1.1.2.3.5 New Tables for External Applications

CONTACT

VIDEO_CONNECTION_TYPE	NOT NULL NUMBER(2)
NO_VIDEO_AVAIL_INDICATOR	NOT NULL NUMBER(1)

EXTERNAL_APPLICATION

VIDEO_CONNECTION_TYPE	NOT NULL NUMBER(2)
NO_VIDEO_AVAIL_INDICATOR	NOT NULL NUMBER(1)

APPLICATION_ROLE_ASSIGNMENT

VIDEO_CONNECTION_TYPE	NOT NULL NUMBER(2)
NO_VIDEO_AVAIL_INDICATOR	NOT NULL NUMBER(1)

2.4.1.1.2.4 PL/SQL Module Definition and Database Trigger Reports

The following PL/SQL modules will be created for CHART R3B3

Travel Time

A PL/SQL module will be written to archive travel time data from the live databases. This will include archiving raw INRIX data, keeping the data for a configurable number of days (e.g., 90 days). This module will also include archiving relevant DMS and/or travel route configuration information, which combined with comm/ops log entries will allow the reporting tool to reconstruct a particular travel time message.

Toll Rates

A PL/SQL module will be written to archive toll rate data from the live databases. This will include archiving raw Vector data, keeping the data for a configurable number of days (e.g., 90 days). This module will also include archiving relevant DMS and/or travel route configuration information, which combined with comm/ops log entries will allow the reporting tool to reconstruct a particular toll rate message.

2.4.1.1.2.5 Database Size Estimate - provides size estimate of current design

Storage of toll rate and travel time raw data for 90 days is expected to increase the size of the live database by up to 10 MB, and increase the size of the archive database by 800 MB/year.

2.4.1.1.2.6 Data Distribution

For R3B3, toll rate data will be obtained from the vector system and stored in a CHART live database prior to being (temporarily) archived. The live database will likely be the AOC database. Travel time data will be obtained from the INRIX system and stored in a live CHART database prior to being (temporarily) archived. This live database will likely be the SOC database.

2.4.1.1.2.7 Database Replication

DMS template information will be distributed to all server sites via database replication.

2.4.1.1.2.8 Archival Migration

The CHART O&M guide contains archive information. In general, the archive process runs nightly to archive and remove data from the live databases. For R3B3 travel time and toll rate data will be archived.

2.4.1.1.2.9 Database Fail-Over Strategy

The SOC is the master database site and the remaining CHART server sites host a snapshot replication database. Should the master database be down or unavailable for an extended period, one of the snapshot databases could be converted to a master database site.

2.4.1.1.2.10 Reports

The CHART database design supports future reporting tool reports that will show travel time and toll rate raw and calculated data to enable the report tool user to decompose a particular travel time/toll rate message for a (configurable) specified period of time (e.g., 90 days).

2.4.1.2 CHART Flat Files

The following describes the use of flat files in CHART.

2.4.1.2.1 Service Registration Files

Each of the CHART background service directories, the JacORB Trader directory, and JacORB Event Service directories has a set of files used to install and uninstall the particular service into the Windows services list. When the service is thus installed it can be controlled through the Windows Services Applet. The files to install and uninstall are `*ServiceReg.cmd` and `*RemoveService.cmd`, where “*” is the name of the service, for instance, HAR or DMS, or HAREvent or DMSEvent (for JacORB event services running for specific CHART services) or Event (for the generic event service used by the GUI and FMS processes) or Trading for the JacORB Trader. These are created at installation time. The registration file is run at installation time, and then these files are not used again. They are merely stored in the unlikely event that they may be needed to re-register the service.

2.4.1.2.2 Service Property Files

Each of the CHART background service directories, the JacORB Trader directory, and JacORB Event Service directories has one properties file used to set runtime parameters used to control execution of the service. These parameters may include location of other services, the database, timeout parameters, retry parameters, etc. These file is named `*.props`, where “*” is the full name of the service, for instance, `HARService`, or `HAREventService` or `TradingService`. These are created at installation time with default values appropriate for most installations. Installation procedures may call for the person performing the installation to edit some files to

make specific updates immediately following installation. These are user-editable ASCII files and parameters are stored in a `Module.ParameterName=value` format, with thorough in-line documentation of each parameter, including defaults and reasonable acceptable ranges and meanings where necessary. Typically only software engineers may occasionally change certain runtime parameters to fine tune performance characteristics.

2.4.1.2.3 GUI Property Files

The CHART GUI has two properties files used to specify runtime parameters. These parameters include location of other services, the database, timeout parameters, retry parameters, etc. The primarily file is named `MainServlet.props`. Additional parameters are stored in the `velocity.props` and `RequestHandler.props` files. These files are stored in the `chartlite` directory under the `WebApps` directory in the Apache Tomcat installation area. These are created at installation time with default values appropriate for most installations. Installation procedures typically call for the person performing the installation to edit some files to make specific updates immediately following installation. These are user-editable ASCII files and parameters are stored in a `Module.ParameterName=value` format, with thorough in-line documentation of each parameter, including defaults and reasonable acceptable ranges and meanings where necessary. Typically only software engineers may occasionally change certain runtime parameters to fine tune performance characteristics.

2.4.1.2.4 Arbitration Queue Storage Files

Each CHART DMS and HAR contains an Arbitration Queue which is used to store and manage the messages requested to be on the online device as part of a response to ongoing traffic events. This data is stored in a file in a directory called `MessageQueuePersist/`, which is a subdirectory of the `DMSService` and `HARService` directories. These are binary files, and are not user-editable or user-viewable from Windows. The files are named by the 32-digit hexadecimal CHART ID plus the extension “.per”. Arbitration Queues are not generally maintained from one version of CHART to the next. Whenever the Java version changes, they cannot be maintained, as the old files will not be readable using the new version of Java.

2.4.1.2.5 Device Logs

DMSs, TSSs, and HARs have a capability to store communications transactions between CHART software and the physical devices over the telephone lines. This data can be used for debugging communications issues or for validating successful communications operations. The device logs can be toggled on or off by editing device properties from the appropriate device details screens. Typically all device communications logging is enabled for all devices. These logs are automatically deleted by the system after a set period of time, so they do not accumulate infinitely. They are stored in the `DeviceLogs/` or `DebugLogs/` subdirectories within the service install directory, and are named by device name and date, plus a “.txt” extension. These logs are typically read only by software engineering personnel.

2.4.1.2.6 Traffic Sensor Raw Data Logs

TSSs are polled periodically (typically every five minutes) for traffic volume, speed, and occupancy data. The statistics gathered are stored in data files in the `TSSService/RawData/`

directory. From here these files are permanently archived for historical purposes. These files are stored in a human-readable, comma-delimited, ASCII format, although they are not designed for convenient routine interpretation directly by users.

2.4.1.2.7 Service Process Logs

All CHART services write to a process log, used to provide a historical record of activity undertaken by the services. These logs are occasionally referenced by software engineering personnel to diagnose a problem or reconstruct a sequence of events leading to a particular anomalous situation. These logs are automatically deleted by the system after a set period of time defined by the service's properties file, so they do not accumulate infinitely. These files are stored in the individual service directories and are named by the service name and date, plus a ".txt" extension. These logs are typically read only by software engineering personnel.

2.4.1.2.8 Service Error Logs

All CHART services write to an error log, used to provide detail on certain errors encountered by the services. Most messages, including most errors, are captured by the CHART software and written to the process logs, but certain messages (typically produced by the Java Virtual Machine itself, by COTS, or DLLs) cannot be captured by CHART Software and instead are captured in these "catch-all" logs. Errors stored in these logs are typically problems resulting from a bad installation; once the system is up and running, errors rarely appear in these error logs. Debugging information from the JacORB COTS, which is not usually indicative of errors, can routinely be found in these error logs, as well. These log files can be reviewed by software engineering personnel to diagnose an installation problem or other type of problem. These logs are automatically deleted by the system after a set period of time defined by the service's properties file, so they do not accumulate infinitely. These files are stored in the individual service directories and are named by the service name and date, plus an ".err" extension. These logs are typically read only by software engineering personnel.

2.4.1.2.9 GUI Process Logs

Like the CHART background services, the CHART GUI service also writes to a process log file, used to provide a historical record of activity undertaken by the process. These GUI process logs are occasionally referenced by software engineering personnel to diagnose a problem or reconstruct a sequence of events leading to a particular anomalous situation. These logs are automatically deleted by the system after a set period of time defined by the GUI service's properties file, so they do not accumulate infinitely. These files are stored in the `chartlite/LogFiles/` directory under the `WebApps/` directory in the Apache Tomcat installation area. They are named by the service name ("chartlite") and date, plus a ".txt" extension. These logs are typically read only by software engineering personnel. Additional log files written by the Apache Tomcat system itself are stored in the `log/` directory in the Apache Tomcat installation area.

2.4.1.2.10 FMS Port Configuration Files

The CHART Communications Services read a Port Configuration file, typically named `PortConfig.xml`, upon startup, which indicates which ports are to be used by the service and

how they are to be initialized. A Port Configuration Utility is provided which allows for addition, removal of ports and editing of initialization parameters. As indicated by the extension, these files are in XML format. This means these files are hand-editable, although the Port Configuration Utility allows for safer, more controlled editing. The Port Configuration files are typically modified only by software engineers or telecommunications engineers.

2.4.2 Database Design

The CHART database design is described below. The design is based on the CHART Business Area Architecture, and the CHART System Requirements.

The database design consists of these major areas:

- User/system management
- Device configuration
- Device status
- Traffic event response planning
- Events and logging
- Alerts
- Notification
- Schedules
- System parameters
- Travel Routes
- Replication
- Archiving

All device configuration data is maintained by the CHART database and is supplied to the FMS as part of a service request. However, configuration data for devices related to video distribution is not supplied to the FMS, since CCTV camera communications do not use the FMS.

2.4.2.1 User/System Management

The user/system management entities consist of the complete suite of information to tie together the users, roles, organizations, and functional rights with the center's identification. The user/system management entities are considered static data in the sense that the majority of the data will be pre-loaded either through a GUI or via SQL loads.

2.4.2.2 Device Configuration

The DMS, HAR, SHAZAM, TSS, Camera, Monitor, and other CCTV video entities include data that define the configuration of the resources for devices. Each device or detector is associated with an organization via a foreign key. The organization is responsible for all devices and for each model type to which it is related.

All of the configuration data is considered static data. It is generally changeable, but changes infrequently.

2.4.2.3 Device Status

The DMS, HAR, SHAZAM, TSS, Camera, and Monitor entities include data that define the status or state of the devices. Some status information (e.g. last poll time, last polled detector speed data) changes very frequently. Other status information (e.g., the message on a DMS) changes less frequently.

2.4.2.4 Traffic Event Response Planning

The planning entity consists of all of the data necessary for an operator to execute a response plan from within an open traffic event. Response plans include preselected HAR and DMS devices with messages related to a well known event such as recurring congestion at a particular location.

This data is considered to be fairly static, although libraries and plans are easily updated. These data set up the plan scenario for a given event. It is used manually by operators to refine the plan or create their own.

The dictionary entity data assists the operator by checking spelling and checking for banned words when creating messages for the message library, for DMS messages, and for HAR text message clips, and by doing pronunciation substitution prior to text to speech for HAR text message clips.

2.4.2.4.1 Events and Logging

The events entity includes all informational data related to traffic incidents. It also includes any devices that are part of the response to an event, such as DMSs and HARs. Also included are various log data that are described in more detail below.

The logs that are maintained are listed below:

- Communications Log
- Event Log
- Operations Log

The Communications Log entity documents operator communications, and may or may not be tied to a specific traffic event. The event log contains operator and system generated entries specific to actions associated with a particular traffic event. The Operations Log entity stores all system generated events, including device usage and component failures.

2.4.2.5 Alerts

The alerts entity includes all informational data related to alerts. Alerts are dynamic data. Most alerts are created by the system automatically, although manually generated generic alerts are also supported. Alert status and history data can be updated frequently. All alert data is archived.

2.4.2.6 Notification

The notification entity includes all informational data related to notifications. Notifications are dynamic data. Notification status data are updated frequently.

2.4.2.7 Schedules

The schedules entity includes all informational data related to schedules. Schedules are fixed data. Users add schedules to the system and delete them when they are done. Schedules do not have dynamic status or history data.

2.4.2.8 System Parameters

The System Profile parameters are used for general CHART system operations. Examples of system parameters include:

- Days to purge operation log
- Which event types may be combined
- Which event types are comparable for event location duplication
- HAR date stamp format
- Alert system configuration parameters
- General GUI parameters

2.4.2.9 Travel Routes

The travel routes entity includes all informational data related to travel routes, used to provide travel time and/or toll rate data for use in traveler information messages. Travel routes are fixed data. Administrators add travel routes to the system in preparation for displaying travel times or toll rates on DMSs. Travel routes do not have dynamic status or history data.

2.4.2.10 Replication

The database will provide replication of all entities required for a CHART server site to run independent of any other CHART server site, as might occur with a network outage between sites. This includes data related to CHART GUI (profile, folders), user management (including external client IDs and public keys), and dictionary data. The data related to logging and resources is replicated as well.

Device configuration data is not replicated since each device is homed to only one server. Other CHART servers access that device configuration through the appropriate CORBA Trading Service. Similarly, traffic event information, alerts information, notification information, and schedule information are homed to only one server and therefore not replicated.

2.4.2.11 Archiving

The CHART Archive database stores data from the CHART operational system as part of a permanent archive. The CHART Archive database design is a copy of the CHART operational system for those tables containing system, alert, and event log information. In addition, the CHART Archive database stores detector data. This data is stored as time annotated averages at selected frequencies. For R3B3, archiving will be updated to include traveler information messages and their underlying data. See Figure 2-6 which includes the ERD for the Archive database.

3 Key Design Concepts

3.1 Travel Routes

As a building block for traveler information messages, R3B3 introduces the concept of travel routes. Each travel route defined in CHART represents a segment of roadway, usually starting at a DMS and ending at some well known point (an exit number, route number, etc.) CHART travel routes are used to supply travel time and/or toll rate data to a DMS for inclusion in a traveler information message. Each travel route may have one or more roadway links included. Each roadway link is identified by an ID, and for R3B3 will always correspond to a link that exists in the INRIX system, which provides travel time data to CHART. A travel route with no roadway links included cannot be used for travel times. Travel routes may also have a toll rate source assigned. Toll rate sources are identified by a beginning and ending ID, and for R3B3 will always correspond to a toll route in the Vector system, which provides toll rate data to CHART. Travel routes without a toll rate source assigned will not be used for toll rates.

In addition to being building blocks for traveler information messages, Travel Routes are also useful by allowing users to view current travel times and toll rates. Sorting and filtering capabilities as well as recent data trends providing users with another means to assess current roadway conditions.

3.2 Traveler Information Message Templates

CHART R3B3 Traveler Information Message Templates are another building block for traveler information messages. An administrator creates templates that specify the layout and content of traveler information messages. These templates are for a specific DMS sign size, and at least one template must exist for each size (rows/columns) of DMS where a traveler information message will be displayed. In addition to text, the content may include data fields, which are place holders within the message where data from travel routes is to be inserted. The following data fields are permitted in R3B3:

- Destination
- Travel Time (actual)
- Travel Time (range)
- Toll Rate
- Distance
- Toll Rate Effective (“as of”) Time

Templates allow the administrator to specify which fields are supplied by the same travel route. This allows templates to contain data from 1 or more travel routes, with the data from each route correlated properly. Templates also allow the administrator to specify the format to be used for each type of data field included in the template. All fields of the same type will share a common format, eliminating the possibility for a mismatch within the same messages. Because it's possible that data fields specified in a template may become unavailable during its actual use, the

administrator also specifies a missing data rule for each template. Using the missing data rule, the administrator can specify if the entire message is invalid if any data is missing, if the page containing the missing data is invalid, or if the row containing the missing data is invalid. The appropriate rule to choose depends on the content and layout of the template.

3.3 Traveler Information Messages

Traveler information messages combine a pre-defined message template with data from one or more travel routes to show motorists current travel times and toll rates on DMSs. Traveler information messages are automatically updated as data from their associated travel routes changes. Traveler information messages are created for any DMS where travel times or toll rates are to be displayed. These messages can be created in advance, and activated by the user when desired. Multiple traveler information messages for a DMS can be created in advance, however only one may be active on a DMS at any given time.

Traveler information messages, when activated, utilize the DMS arbitration queue. Two new “buckets” created for R3B3 are used to set the initial queue priority for toll rate and travel time messages. Any traveler information message that contains toll rate data is considered a toll rate message (even if it also contains travel time data) and will initially be placed in the “toll rate” queue bucket. Any traveler information message that contains travel time data (but not toll rate data) is considered a travel time message and will be initially placed in the “travel time” queue bucket. The system allows the administrator to override this behavior per DMS and specify different buckets to be used for toll rate and travel time messages. Once a traveler information message is on a DMS arbitration queue, all existing arbitration queue features apply, including the ability to reprioritize the message within the queue, and the ability to combine the message with other messages on the queue (if so configured).

CHART R3B3 includes a travel time display schedule which specifies the periods during the day when travel time messages may be displayed. Travel time messages may be activated or remain active on the arbitration queue during times when travel times are not scheduled to be displayed, however the message will only be shown on the DMS during times when travel time display is scheduled. The system-wide travel time display schedule can be overridden per DMS.

3.4 External Interface to INRIX

For CHART R3B3, an external interface to the INRIX system is used to obtain travel times for travel routes. INRIX is a web service available on the Internet, and CHART will connect to it periodically (on the order of every 5 minutes) via HTTPS to obtain travel time data for roadway links within Maryland. Once travel time data is obtained from INRIX, CHART will update the travel time data for roadway links used by CHART travel routes, and the travel routes will update their overall travel time by adding together the travel times of each link contained in the travel route. Changes to travel time data for a travel route propagate within CHART to any active traveler information message using data from that travel route.

Settings in CHART allow a percentage of a link to be used when computing the overall travel route travel time to accommodate situations such as when locations of DMSs that will display travel times do not match cleanly to INRIX link starting points. Other settings in CHART specify the minimum data quality CHART will accept from INRIX, and CHART travel routes

will consider the travel time unavailable if too many links fall below their configured minimum quality level.

3.5 External Interface to Vector

An external interface to the MdTA Vector system is used by CHART R3B3 to obtain current toll rates for CHART travel routes. Toll rate data is pushed to the CHART system via a web service interface included in R3B3. The Vector system connects to this service via HTTPS and supplies data to CHART in an XML format. When the CHART web service receives data from the Vector system, it updates the current toll rate data for any CHART travel route that has a toll rate source specified and that source is the Vector system (Vector is the only toll rate source for R3B3). Changes to the toll rate for a CHART travel route are propagated to any traveler information message that includes toll rate data from that travel route.

3.6 External Interface to RITIS

For CHART R3B3, the existing external interface to RITIS is enhanced and expanded. The existing traffic event import from RITIS is being enhanced to provide configurable filtering, alerts, and flagging of interesting events. The RITIS interface is being expanded to allow DMS and TSS devices included in RITIS to be shown within the CHART system. The transport layer used for these RITIS connections continues to be RITIS-specific using Apache's ActiveMQ implementation of the Java Messaging Service (JMS). The traffic event data layer implements the J2354 standard with a few RITIS extensions to the standard. The DMS and TSS data layers implement the TMDD standard for each of these devices. The sections below describe further concepts related to RITIS data importing for R3B3.

Geographical Areas

CHART R3B3 introduces the concept of geographical areas to allow geographic filtering of traffic events, DMSs, and TSSs that exist in the RITIS system. An administrator can define a geographical area as a polygon containing 3 or more points specified by latitude/longitude. The system allows the administrator to upload these points from a KML file rather than typing them by hand to avoid typographical errors and to allow an external mapping product to be used to define the areas. R3B3 provides features to manage these geographical areas, including add, edit, delete, and view.

Traffic Event Import Rules

CHART R3B3 includes the ability to define traffic event import rules that will be applied automatically by the existing RITIS traffic event importer. Each rule can contain one or more filter criteria, including geographical areas, route types, number of lanes closed, event type, and search type. A RITIS event must match all criteria specified in a rule to match the rule, and must match at least one rule to be imported into CHART. Each import rule can also contain rule actions. These actions will be performed when a RITIS event matches the rule (and is therefore being imported into CHART). The available actions for R3B3 are Issue Alert, Send Notification, and Mark as Interesting. The Issue Alert action, when enabled, causes CHART to send an alert to a specified operations center when an event is imported that matches the rule. The Send Notification action, when enabled, causes CHART to send a notification to a specified

notification group when an event is imported that matches the rule. The Mark as Interesting action, when enabled, causes CHART to set the “interesting” flag for the event when an event is imported that matches the rule. The “interesting” flag causes the event to appear on the home page of the CHART GUI in the external events tab.

Create CHART Event from RITIS Event

CHART R3B3 adds the ability to create a new CHART event using data from an external event (imported from RITIS). When this is done, the user can edit the basic event and location data prior to saving the new CHART event. The system automatically associates the new CHART event with the external event from which the operation was initiated.

DMS and TSS Import

CHART R3B3 includes the ability for the administrator to include view-only copies of RITIS DMS and TSS devices in the CHART system. Once added to the CHART system, the CHART RITIS import service will keep the status of these devices updated within CHART. The CHART system allows the administrator to specify which devices from RITIS are to be included in CHART, as well as those the administrator wants excluded from CHART. A query capability that includes several search criteria allows the administrator to search the potentially large list of DMS and TSS devices that may exist in RITIS so that they may choose devices to include in CHART or mark as excluded. The search feature also allows the administrator to view the devices they have already marked as included, already marked as excluded, and/or not yet marked as either included or excluded.

The existing CHART DMS and TSS lists are enhanced in R3B3 to allow users with rights to view external DMS or TSS to show or hide these external devices within the lists. When external devices are shown, the system allows the user to filter the list to hide CHART devices and to filter the list to show only external devices from specific agencies. CHART R3B3 uses a different background color to differentiate external devices from CHART devices within device lists. The device details pages for external devices are read-only for all users, except that privileged users are permitted to mark the external device as “excluded”, removing it from the CHART system.

Archiving of External DMS and TSS Data

External DMSs and TSSs that have been imported into the CHART system will be archived for offline analysis along with internal CHART DMSs and TSSs, and will be permanently flagged as external devices in the archive.

Owning Organizations

When importing traffic events, DMS devices, and TSS devices into CHART, R3B3 will utilize a mapping from external system / agency to a CHART organization. If a mapping is not found for a traffic event or device that is imported, the system will use a default organization.

3.7 CHART Data Export

CHART R3B3 includes a web service that allows pre-approved external systems to obtain data from the CHART system. External systems can issue data requests (via HTTPS) and receive the requested data in the form of an XML document. R3B3 allows traffic events, DMS, TSS, HAR,

and SHAZAM data to be retrieved in this manner. Authentication and data protection schemes (as described in section 2.3 above) ensure that only authorized clients can retrieve data, and that clients can only retrieve data for which they are permitted to receive.

3.8 External System Connection Status

CHART R3B3 includes the ability to view the status of all external connections, including those connections from CHART to RITIS and INRIX, and connections to CHART from Vector and clients using the CHART export web service. R3B3 also allows the administrator to configure alerts and notifications for each external connection. The administrator can configure the system to alert a specified operations center when a connection failure is detected. A notification group can also be specified to receive a notification when a connection failure is detected. Optionally, the administrator can also configure the system to send an alert or notification when warning conditions are detected. The system utilizes an administrator specified threshold time to prevent a flood of alerts and/or notifications from being sent if a connection is in a state where it is frequently transitioning between OK and Failed.

3.9 Device Locations

CHART R3B3 updates all devices (except monitors) to include location fields. The following fields are added:

- State
- County
- Route Type
- Route
- Direction
- Intersecting Feature (state milepost or intersecting route)
- Latitude / Longitude

Device lists are updated to include columns for County, Route, and Direction, and to allow the user to sort and filter on these new columns.

CHART R3B3 uses device latitude/longitude (when specified) and traffic event latitude/longitude (when present) to allow users to view devices close to a traffic event. The user can select a radius when viewing the traffic event details to see all devices within that radius. The user can also directly add DMS and HAR devices to the response plan of the traffic event when they appear in the list of devices close to the event. External DMSs and TSSs that are included in the CHART system and are within the specified radius from the event will appear in the list of close devices; however the user cannot add external DMSs to an event response plan.

3.10 TCP/IP DMS and TSS Communications

CHART R3B3 adds TCP/IP as a new communication option for DMS and TSS devices. When the physical device supports this option, and the device is configured within CHART to use

TCP/IP communications, the CHART backend service will communicate directly with the device over the network and will not utilize a port manager as it does for POTS, ISDN, and direct connect RS232 communications. TCP/IP device communications will bypass the FMS servers and the CHART Communications Services entirely.

3.11 Error Processing

In general, CHART traps conditions at both the GUI and at the server. User errors that are trapped by the GUI are reported immediately back to the user. The GUI will also report communications problems with the server back to the user. The server may also trap user errors and those messages will be written to a server log file and returned back to the GUI for display to the user. Additionally, server errors due to network errors or internal server problems will be written to log files and returned back to the GUI.

3.12 Packaging

This software design is broken into packages of related classes. The table below shows each of the CHART packages along with a description of each. New for R3B3 are the following packages: TravelRouteModule, MessageTemplateModule, Travel Time Import Module, DMSImportModule, TSSImportModule, dataCache servlet, CHARTExport servlet, and the Toll Rate servlet.

Table 1 Package Descriptions

Package Name	Package Description
ActionUtility	This package contains code used by the GUI to invoke actions from alerts generated by the Schedule Module. This package is separate from the Schedule Module itself because it is currently used by the GUI, but may be used by the Schedule Module itself in future releases.
AlertModule	This package contains an installable service application module that is responsible for handling Alerts in CHART. This module will change for R3B3 to support new alert types.
AudioClipModule	This package contains classes used during the creation and storage of HAR audio clips.
AudioCommon	This contains the CORBA interfaces, structs, enums, and constants used to define the interface between the CHART AudioClipModule and other applications such as the CHART GUI.
Camera Control Module	This package contains an installable service application module that serves the Camera Factory, Camera and related objects as specified in the system interfaces. This module will change for R3B3 to support location data for cameras.
chartlite	This package contains all of the classes that comprise the CHART GUI.
CHART2Service	This package contains a class that serves as a generic service application.
CommandProcessorModule	This package contains an installable service application module that serves the CommandProcessorFactory, CommandProcessor and related objects as specified in the system interfaces.
CommLogModule	This package contains classes that are used to write the CommunicationsLog.

Package Name	Package Description
CORBAUtilities	This package contains classes included in the third party ORB product used for implementation.
CHARTExport servlet	The CHART Export servlet is used to handle those clients who wish to receive data from CHART.
DataCache servlet	The DataCache servlet is used to cache the data that will be sent to client who wish to receive data from CHART.
DataModel	This package contains classes and methods that allow for storage, efficient lookup, and updating of object data.
DeviceManagement	This package contains the CORBA interfaces, structs, enums, and constants used to define the interface between the CHART ArbitrationQueue and other applications such as the CHART GUI. This package will change for R3B3 to support device locations and changes to the arbitration queue to support traveler information messages.
DeviceUtility	This package contains various utility classes used by CHART devices.
DictionaryManagement	This package contains the CORBA interfaces, structs, enums, and constants used to define the interface between the CHART Dictionary and other applications such as the CHART GUI.
DictionaryModule	This package contains an installable service application module that serves Dictionary and related objects as specified in the system interfaces.
DMSControl	This package serves the DMS Configuration and Status Factory, DMS Configuration and Status and related objects as specified in the system interfaces. This will change for R3B3 to support device locations, TCP/IP communications, and NTCIP DMS font settings.
DMSControlModule	This package contains an installable service application module that serves the DMS Factory, DMS and related objects as specified in the system interfaces. This will change for R3B3 to support device locations, TCP/IP communications, and NTCIP DMS font settings.
DMSProtocols	This package contains classes that encapsulate the functionality used to communicate with the various models of DMSs
DMSUtility	This package contains DMS related utility classes used by the server.
DMSImportModule	This package contains an installable service application module that is used to import external DMS data into CHART.
EventImportModule	This package contains an installable service application module that is used to import external traffic event data into CHART.
ExternalInterfaceModule	This package implements connections to external systems. Currently RITIS is the only external system connecting to CHART. !
EORS	This package contains classes related to EORS.
EORSModule	This package contains an installable service application module that serves EORS and related objects as specified in the system interfaces.
FieldCommunicationsModule	This package contains an installable service application module that serves Port manager and related objects used to provide access to communications ports on the machine where this module is run.
GeoAreaModule	This package contains an installable service application used for managing and providing access to Geographical Areas configured in CHART.

Package Name	Package Description
HAR Control	This package contains HAR utility and other HAR related classes. This package is changed in R3B3 to include device locations.
HARControlModule	This package contains an installable service application module that serves the HAR Factory, HAR and related objects as specified in the system interfaces. This package is changed in R3B3 to include device locations.
HARProtocols	This package contains classes that encapsulate the functionality used to communicate with the various models of HARs.
LogCommon	This package contains objects related to the commLog.
MessageLibraryModule	This package contains an installable service application module that serves the MessageLibrary Factory, MessageLibrary and related objects as specified in the system interfaces.
MessageTemplateModule	This package contains an installable service application module that serves the MessageTemplate Factory, and related objects as specified in the system interfaces.
MonitorControlModule	This package contains an installable service application module that serves the Monitor Factory, Monitor and related objects as specified in the system interfaces.
NativeUtility	This package contains utility classes used for calling C++ code.
Notification Module	This package contains an installable service application module that provides notification services for CHART.
PlanModule	This package contains an installable service application module that serves the Plan Factory, Plan and related objects as specified in the system interfaces.
ResourcesModule	This package contains an installable service application module that serves the OperationsCenter Factory, OperationsCenter and related objects as specified in the system interfaces.
RoadwayLocationLookupModule	This package contains an installable service application module that provides interfaces for querying the location data contained on the CHART Mapping database.
RouterControlModule	This package contains an installable service application module that serves the Router Factory, Router and related objects as specified in the system interfaces.
ScheduleModule	This package contains an installable service application module that serves the Schedule Factory and Schedule objects as specified in the system interfaces. This package is changed in R3B3 to include device locations.
SHAZAMControlModule	This package contains an installable service application module that serves the SHAZAM Factory, SHAZAM and related objects as specified in the system interfaces. This package is changed in R3B3 to include device locations.
SHAZAMProtocols	This package contains classes needed for communication to a specific model SHAZAM.
SHAZAMUtility	This package contains SHAZAM related utility.
TollRateImport servlet	The TollRateImport servlet provides web service to allow data providers (Vector) to import toll rates into Chart.

Package Name	Package Description
TrafficEventMangement	This package contains classes related to TrafficEvent objects.
TrafficEventModule	This package contains an installable service application module that serves the TrafficEvent Factory, TrafficEvent and related objects as specified in the system interfaces.
TravelRouteModule	This package contains an installable service application module that serves the Travel Route Factory, and related objects as specified in the system interfaces.
TravelTimeImportModule	This package contains an installable service application module that serves the INRIX Import related objects as specified in the system interfaces
TSSImportModule	This package contains an installable service application module that is used to import external TSS data into CHART
TSSMangementModule	This package contains an installable service application module that serves the RTMS Factory, RTMS and related objects as specified in the system interfaces. This package is changed in R3B3 to include device locations, TCP/IP device communications.
TSSUtility	This package contains TSS related utility classes.
TTSCControlModule	This package contains an installable service application module that is used to run the TTS server. This package is changed in R3B3 to include device locations, TCP/IP device communications.
Utility	This package contains various utility classes used throughout CHART.
VideoSwitchControlModule	This package contains an installable service application module that serves the VideoSwitch Factory, VideoSwitch and related objects as specified in the system interfaces.
VideoUtility	This package contains Video related utility classes.

3.13 Assumptions and Constraints

1. Knowledgeable SHA, MdTA, RITIS, INRIX, and VECTOR subject matter experts will be available to support project schedules.
2. The design of the INRIX system interface and the data provided by INRIX will be substantially the same as documented in the “I-95 Vehicle Probe Project Interface Guide”, version 2.2 dated October 11, 2008.
3. The design of the VECTOR system interface will be substantially the same as documented in the Toll Rate Vector to CHART Interface Control Document, Revision 1, document number WO15-ID-001R1, dated November 25, 2008.
4. A connection to the INRIX system has been provided to the development team. However, the development team has not yet been tested this connection.
5. We assume that a connection to the VECTOR system will be provided to the development team sometime in the early implementation phase of R3B3.
6. CHART will assume INRIX travel time data is good – correct and smoothed. CHART cannot validate INRIX data, and will not smooth it. No limit on route travel times variance from one moment to the next will be imposed.
7. CHART will assume VECTOR toll rate data is correct. No minimum/maximum limit on toll rates will be imposed. CHART cannot validate VECTOR data. For instance, CHART will not understand the relationship between toll rate routes, and therefore cannot validate that a longer route should have a higher toll than a shorter route.

8. In general, once configured and enabled, travel times and toll rates will be posted to DMSs automatically. This breaks a long-standing philosophy that all messages will be approved by a human operator before being posted.
9. Travel times are not intended for incident detection. Travel times should not be used for automatic incident detection, and will not be presented in a manner geared to facilitate operator-based incident detection.
10. Testing of DMS and detectors connected to the system via TCP/IP will be done using simulators. If actual devices are made available by SHA prior to the start of integration test they will be used for testing. If actual devices are not provided for testing during integration test, testing with live devices will occur after R3B3 is deployed, and any post deployment changes will be billed under a work order and will not fall under warranty.
11. Testing of the change to set the NTCIP DMS font and line spacing will primarily be done using simulators. If an actual NTCIP DMS is made available by SHA prior to the start of integration test it will be used for integration testing. If an actual NTCIP DMS is not provided for testing during integration test, testing with a live NTCIP DMS should be performed instead. If there is no available NTCIP DMS to use for testing, then the testing will occur after R3B3 is deployed, and any post deployment changes will be billed under a work order and will not fall under warranty.
12. SHA personnel will populate the new device location fields after R3B3 is deployed.
13. CHART R3B3 will use the existing NTCIP DMS implementation as is or slightly modified, to command and control toll rate signs. This system is a robust system capable of handling many but not all problems and has been designed to have an uptime and (device command and control, etc.) accuracy sufficient and acceptable for CHART traffic operations. It has not been, nor will it be for R3B3, a system with an uptime and/or accuracy approaching that required for a true financial system. There are planned and unplanned outages, device nuances and failures, etc. that may occur with or without knowledge and/or participation by CSC/Team CSC staff. These and other events may impact CHART R3B3's ability to obtain and/or correctly display toll rates.

Should problems occur, CSC's responsibility shall remain to fix and/or enhance the system according to the problem report (PR) process (i.e., document the problem via PR and schedule the fix in a patch or release to the software) or according to the normal warrantee provisions. Any and all responsibility for resolving financial disputes related to toll rate signs, including payments due to incorrect/inconsistent toll rates, shall be the responsibility of the corresponding State entity, and not CSC.

4 Use Cases

The use case diagrams depict new functionality for new CHART R3B3 features.

4.1 High Level

4.1.1 R3B3HighLevel (Use Case Diagram)

This use case diagrams shows use cases related to new R3B3 features and enhancements to existing features at a high level.

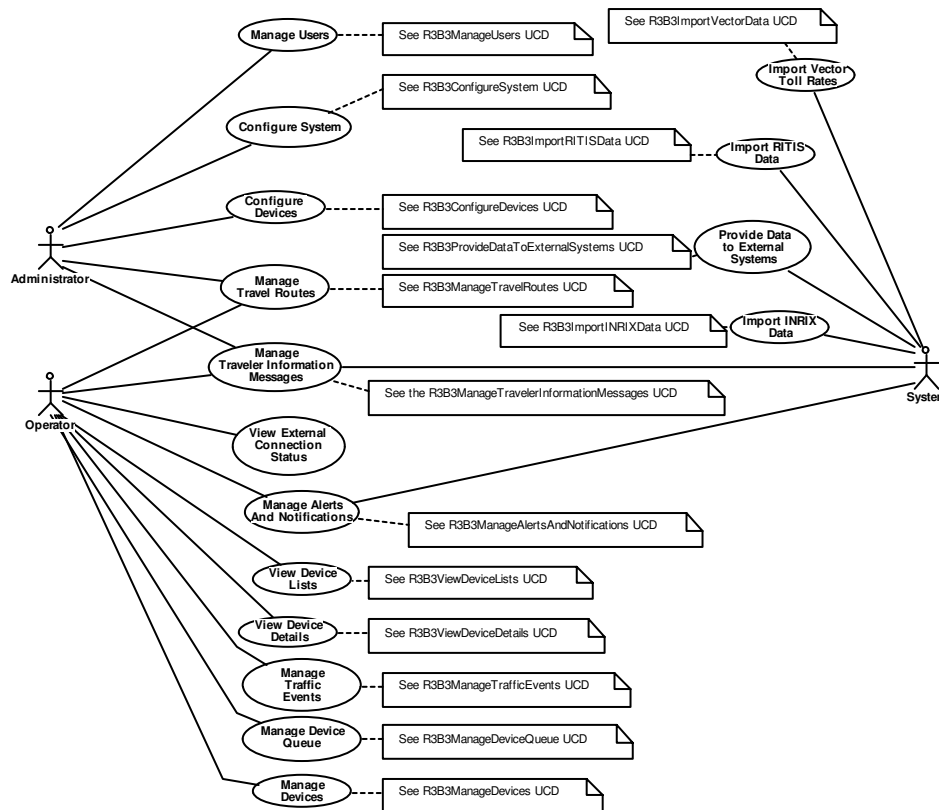


Figure 4-1. R3B3HighLevel (Use Case Diagram)

4.1.1.1 Administrator (Actor)

An administrator is a CHART user that has functional rights assigned to allow them to perform administrative tasks, such as system configuration and maintenance.

4.1.1.2 Configure Devices (Use Case)

An administrator (operator with the correct functional rights) may configure devices. This includes the devices themselves and all associated supporting configuration information.

4.1.1.3 Configure System (Use Case)

An administrator can edit the system configuration via the system profile.

4.1.1.4 Import INRIX Data (Use Case)

The system shall connect to the INRIX system and periodically import travel time data for links on a configurable interval. CHART will maintain the connection by periodically obtaining a new authentication key as required by the INRIX system. An External Connection Alert (when enabled for the INRIX connection) will be sent to the operations center configured to receive alerts for the INRIX connection. The following conditions will trigger an alert: Data cannot be retrieved from INRIX for a configurable period of time Data retrieved from INRIX does not comply with the documented format Data retrieved from INRIX does not contain data for a link that is included in a CHART travel route

4.1.1.5 Import RITIS Data (Use Case)

The system shall import data from the RITIS system.

4.1.1.6 Import Vector Toll Rates (Use Case)

4.1.1.7 Manage Alerts And Notifications (Use Case)

A user with proper functional rights can view and respond to alerts generated by the system. The system will monitor conditions and send out alerts and/or notifications. For R3B3 three new alert types are being added: Travel Time Alerts, External Connection Alerts, and External Event Alerts. The conditions that trigger these alerts can also cause a notification to be sent independent of the alert. Details are shown in the Manage Alerts And Notifications use case diagram.

4.1.1.8 Manage Device Queue (Use Case)

Users with appropriate privileges can manage device queues. The traveler information messages added to R3B3 utilize the DMS message queue and two new priorities are added to support these messages. See the ManageDeviceQueues UCD for details.

4.1.1.9 Manage Devices (Use Case)

An operator with the correct functional rights may perform basic operations on CHART devices including HARs, DMSs, Video related devices, TSSs, and SHAZAMs. For R3B3, DMS and TSS communications will support TCP/IP (in addition to existing communication methods), and the communications to set messages on NTCIP DMSs will be enhanced to set the font and line spacing. See the R3B3ManageDevices UCD for details.

4.1.1.10 Manage Traffic Events (Use Case)

This diagram models the actions that an operator may take that relate to traffic events. This

includes responding to traffic events using field devices.

4.1.1.11 Manage Travel Routes (Use Case)

An administrator will be able to manage the configuration of travel routes in the system. Travel routes represent a section of roadway (not necessarily on the same roadway/signed route) for which travel time and/or toll rate information may be provided. The administrator can add, edit, or remove travel routes. An operator will be able to view the currently defined toll routes and the travel time / toll rate information for those routes. See the Manage Travel Routes use case diagram for details.

4.1.1.12 Manage Traveler Information Messages (Use Case)

The administrator will be able to define traveler information DMS message templates for building traveler information messages (toll rate and/or travel time messages). An administrator will set up the messages for a DMS using the templates (see the Configure Devices diagram) and then the operator will be able to enable or disable the messages. Once a message is enabled, the system will format a message for use by a DMS, add it to the arbitration queue, and keep the message up to date when a travel route's data changes. See the Manage Traveler Information Messages diagram for details.

4.1.1.13 Manage Users (Use Case)

An administrator can manage the users that are given access to the system and manage the roles and rights that are used to specify the actions a user may perform and the data they may access. In R3B3, new rights are added for new features, and rights are also being added to provide better control of sensitive data. See the R3B3ManageUsers UCD for details.

4.1.1.14 Operator (Actor)

An operator is a user of the system who has been assigned a valid username/password combination and granted roles for system access.

4.1.1.15 Provide Data to External Systems (Use Case)

The system shall provide access to external systems via a web service to allow them to receive data that the CHART system makes available to third parties. One or more Roles assigned to each external client will be used to determine the data the client will be permitted to access. All requests made by external systems shall be validated against published XSD. CHART will return a response XML document for each request. The XML returned will contain an error code and error text for invalid requests, and will return the requested data for valid, authorized requests. The response XML shall be formatted as specified in published XSD.

4.1.1.16 System (Actor)

The System actor represents any software component of the CHART system. It is used to model uses of the system which are either initiated by the system on an interval basis, or are an indirect by-product of another use cases that another actor has initiated.

4.1.1.17 View Device Details (Use Case)

The system allows users to view details pertaining to devices. In R3B3 several new details are being added, such as device locations. See the R3B3DeviceDetails UCD for details.

4.1.1.18 View Device Lists (Use Case)

The system allows users to view lists of devices that exist in the CHART system. For R3B3 several new data fields are being added to most device lists, such as location data. See the R3B3ViewDeviceLists UCD for details.

4.1.1.19 View External Connection Status (Use Case)

The system shall maintain the state of external connections established by the system and allow users to view the status of the connections, including an indication of any connections that are detected to be down. The following data will be displayed for each connection: connection name, current status (OK, WARNING, FAILED), descriptive status text (optional for status of OK), the time the connection transitioned to its current status, and the time the status was last confirmed.

4.2 Travel/Toll Routes

4.2.1 R3B3ManageTravelRoutes (Use Case Diagram)

This diagram shows use cases related to managing travel routes.

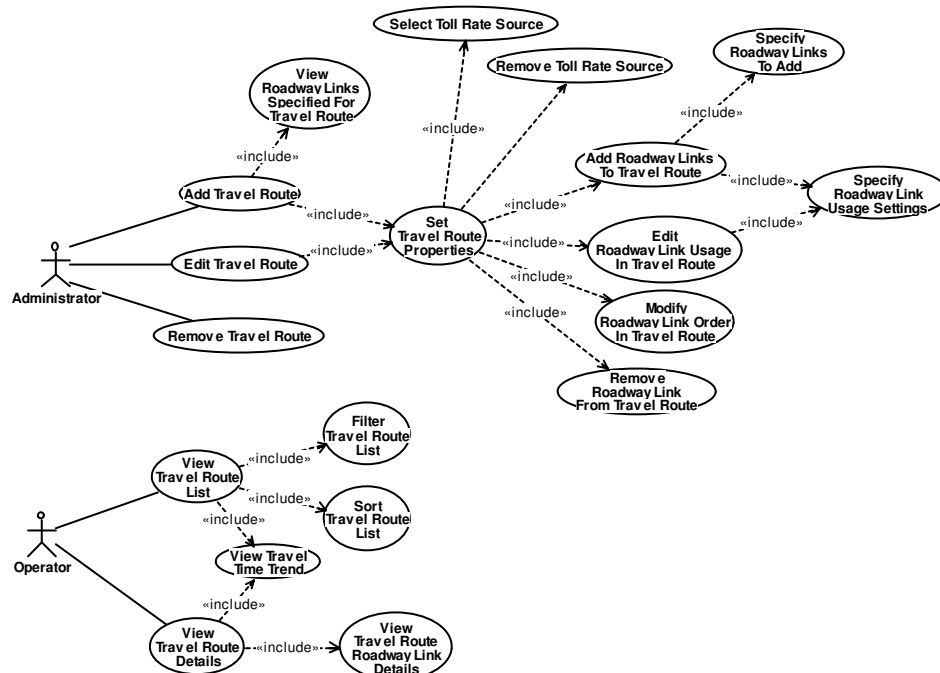


Figure 4-1 R3B3ManageTravelRoutes (Use Case Diagram)

4.2.1.1 Add Roadway Links To Travel Route (Use Case)

A user with appropriate rights shall be permitted to add a roadway link to a travel route.

4.2.1.2 Add Travel Route (Use Case)

The system shall allow a user with appropriate rights to add a travel route to the system.

4.2.1.3 Administrator (Actor)

An administrator is a CHART user that has functional rights assigned to allow them to perform administrative tasks, such as system configuration and maintenance.

4.2.1.4 Edit Roadway Link Usage In Travel Route (Use Case)

The system shall allow a user with appropriate rights to edit the configuration data for a link that has been previously added to a travel route. For details see the

SpecifyRoadwayLinkUsageSettings use case.

4.2.1.5 Edit Travel Route (Use Case)

The system shall allow a user with appropriate rights to edit the configuration of a travel route that exists in the system.

4.2.1.6 Filter Travel Route List (Use Case)

The user shall be able to filter the travel route list based on the value of a specific field (or fields). The sortable fields include: travel time (based on hard-coded ranges), travel time trend (up/down/flat), average speed (using hard-coded ranges), toll rate, roadway route, travel direction, and county.

4.2.1.7 Modify Roadway Link Order In Travel Route (Use Case)

The system shall allow a user with appropriate rights to modify the order of roadway links used in a travel route by moving a roadway link up or down in the order.

4.2.1.8 Operator (Actor)

An operator is a user of the system who has been assigned a valid username/password combination and granted roles for system access.

4.2.1.9 Remove Roadway Link From Travel Route (Use Case)

The system shall allow a user with appropriate rights to remove a roadway link from a travel route.

4.2.1.10 Remove Toll Rate Source (Use Case)

A user with sufficient rights will be able to remove a toll rate source from a travel route, effectively disabling toll rates for the travel route.

4.2.1.11 Remove Travel Route (Use Case)

A user with appropriate rights shall be permitted to remove a travel route from the system. The system will prompt the user for confirmation and will provide a warning if the travel route is being used by one or more DMSs.

4.2.1.12 Select Toll Rate Source (Use Case)

A user with sufficient rights will be able to select a toll rate source for the travel route.

4.2.1.13 Set Travel Route Properties (Use Case)

The system shall allow a user with appropriate rights to set the properties associated with a

travel route. A travel route has a name, up to three destination names (which the system will ensure are of different lengths, for use on different sizes of DMSs), location settings, and travel time and/or toll rate settings. The location settings, which can be derived from the links if the route supports travel times, may also be specified manually, and these settings include: counties; route type, route, and direction for each route in the travel route; travel route length. (The system will require the user to specify the location settings). If the travel route supports toll rates, the user will be able to specify the toll rate source and enable or disable the toll rate functionality. If a route supports travel times (i.e., with one or more roadway links), the user can enable or disable the travel time functionality. A travel route supporting travel times will require a minimum displayable travel time to be specified (below which the displayed travel time will be constrained to the minimum) and a maximum travel time (above which the travel time will not be displayed). A travel route supporting travel times will include a setting for the number of links in the route that are allowed to fall below the per-link quality threshold for the route travel time to be used. There will also be a time threshold that, if exceeded, can trigger a Travel Time Alert sent to a specified operations center and/or notification sent to a specified notification group. The user will be able to enable or disable travel time alerts and notifications after they are set up. Similarly the user can enable/disable toll rate alerts and notifications for travel routes that support toll rates. Toll rate alerts are sent when a toll rate expires.

4.2.1.14 Sort Travel Route List (Use Case)

The travel route list shall permit the user to sort the list based on specific fields, in ascending and descending order. The fields include: travel route name, length, travel time, travel time trend, current speed, toll rate, the name of the DMSs using the route, the roadway route, direction, and county.

4.2.1.15 Specify Roadway Link Usage Settings (Use Case)

A user with appropriate rights shall be permitted to set the properties of a link associated with a travel route. These settings include the percentage of the link to include, and the minimum travel time quality to accept (low, medium, or high).

4.2.1.16 Specify Roadway Links To Add (Use Case)

The system shall allow the user to select one or more links to add to a travel route. The user will be able to select from among a list of all links in the system. To reduce the number of links to choose from, the user will be able to filter the list of links by county, road type, road, direction, and external ID. After adding the first link, the user will be able to display a list of suggested links for selection. The system will use the latitude / longitude of the endpoints to determine what other links are nearby, and may also use the previous link's attributes for filtering. When displaying the links available for selection, the system will display the details for each link including the name of the external system, the external link ID, the link name, route name and/or number, travel direction, link length, county, and (for all links except the first) the distance of the starting point to the previous link's endpoint.

4.2.1.17 View Roadway Links Specified For Travel Route (Use Case)

The system will display the details for roadway links that have been specified to be part of the travel route when adding a travel route to the system. Details will include the external system name, the external link ID, the name of the roadway link, the route name and/or number, the travel direction, the county, the length of the link, and the distance of the link's starting point from the prior link's endpoint (for all links except the first one).

4.2.1.18 View Travel Route Details (Use Case)

A user with appropriate user rights shall be permitted to view the details for a travel route. The details will include current status information (if available/applicable) including the travel time, travel time trend, speed, and toll rate. Details will be displayed for each roadway link including the external link ID, link name, travel time, travel time trend, and current speed (as reported by the data source, if available; otherwise calculated from travel time and link length). Recent history will be shown for each link for each 5 minute period in the last hour, and will include the reported travel time (minutes and seconds) and data quality. If there are multiple samples in a 5 minute period, the latest will be shown. Configuration details will also be shown for each link including the name of the external system, the external link ID, the link name, route name and/or number, traffic direction, length, county, and distance from the prior link (if any). The route details will include details for a toll rate source (if applicable), which includes the recent toll rate history in 5-minute increments for the last hour. (If more than one toll rate is reported in a 5-minute increment the latest will be used). The toll rate source configuration will also be displayed including the external system name, the start ID, the end ID, and the source description. The route details displayed will include up to three destination names of varying lengths. The route's location information will be displayed including: county(ies), roadway route name/number and travel direction (for each roadway route), travel route length, and an indicator showing whether the location was manually entered or derived from links. The travel time settings will be displayed including: a travel time enabled/disabled indicator, minimum/maximum usable travel time, alert/notification travel time, alert enabled/disabled flag and operations center, notifications enabled/disabled flag and notification group, and the number of allowable links below the quality threshold. The toll rate settings shall be displayed and shall include the toll rates enabled/disabled indicator, the operations center that is to receive toll rate alerts (if any), and the notification group to receive toll rate notifications (if any).

4.2.1.19 View Travel Route List (Use Case)

Users with appropriate user rights shall be permitted to view the travel routes defined in the system. The data shown for each route shall include the name of the travel route, length, travel time (if applicable), travel time trend indicator (if applicable), average speed (total route length divided by travel time), toll rate (if applicable), DMSs using the route (i.e., configured to possibly display messages involving the route, even if not currently

enabled/active), roadway route(s), direction(s), and county(ies). The list of routes will allow the user to choose which column to display, and it will allow sorting and filtering.

4.2.1.20 View Travel Route Roadway Link Details (Use Case)

A user with sufficient user rights shall be permitted to view the details of a roadway link included in a route. The details will include status information including the current travel time, average speed, travel time trend, and last status time. It will also include the recent travel time history for the last hour in 5-minute increments (including the latest of the samples if more than one falls in a 5-minute increment) and the reported quality of each sample. The configuration for the link will be displayed including the external system name, external link ID, link name, county(ies), route type, route, travel direction, link length, and geographic coordinates of the starting / ending point.

4.2.1.21 View Travel Time Trend (Use Case)

The system will display the travel time trend. The trend is computed by comparing the average of the latest N recent travel times to the average of the earliest N travel times, where N is a configurable system-wide travel time sample size, the number of recent travel times to compare. The Trend shall be considered "up" if the recent history contains at least 2N travel times and the average of the N latest travel times is at least X percent greater than the N earliest travel times, where X is the system-wide travel time threshold. The trend shall be considered "down" if the history contains at least 2N travel times and the average of the N latest travel times is at least X percent less than the average of the N earliest travel times. The trend shall be flat if the recent history has less than 2N entries or the average of the N latest travel times is within X percent of the N earliest travel times.

The system shall retain at most 12 recent travel times that have been computed within the last hour. The user will be able to view the travel times that were used to compute the trend.

4.2.2 R3B3ImportInrixData (Use Case Diagram)

This diagram shows use cases related to the import of travel time data from the external INRIX system.

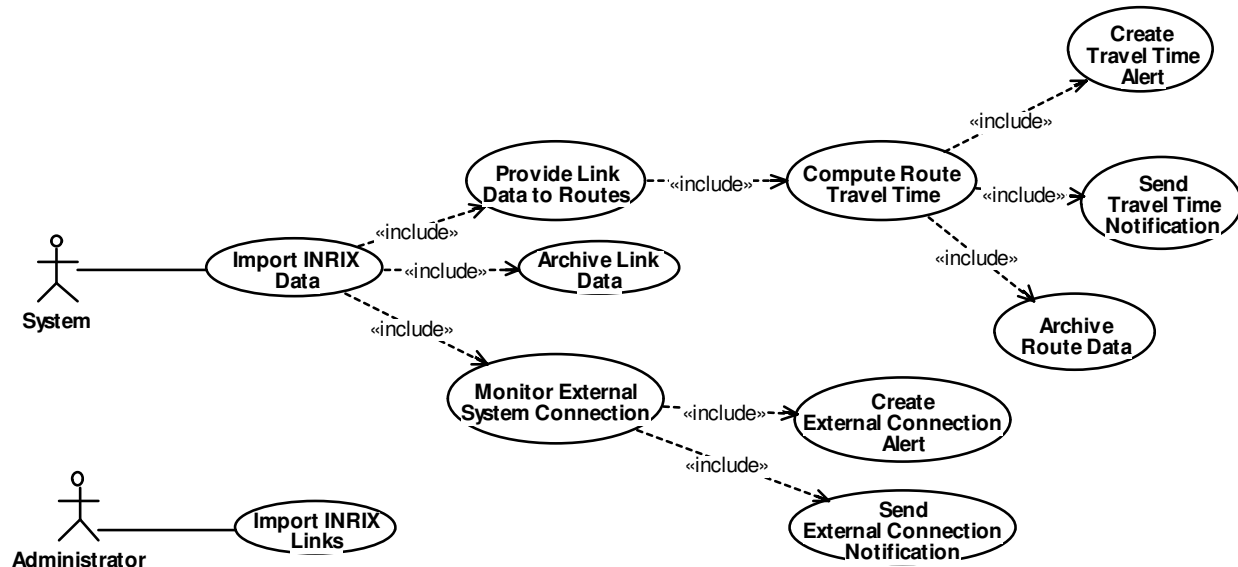


Figure 4-2 R3B3ImportInrixData (Use Case Diagram)

4.2.2.1 Administrator (Actor)

An administrator is a CHART user that has functional rights assigned to allow them to perform administrative tasks, such as system configuration and maintenance.

4.2.2.2 Archive Link Data (Use Case)

The system shall archive all raw link data retrieved from INRIX for a configurable period of time.

4.2.2.3 Archive Route Data (Use Case)

The system shall store data in a form suitable for archival purpose, shall mark data when it is ready to be archived, and shall archive and delete marked archivable route data.

4.2.2.4 Compute Route Travel Time (Use Case)

The system will compute the travel time for a route if the travel time data is available for all links in the route and the number of links having data less than the acceptable quality (as specified in the link's usage settings) is not greater than the maximum number links allowed to have poor quality data (as specified in the travel route settings). Otherwise, if travel time data is missing for any of the links or the quality is low for too many links, the travel time

data for the route will be unavailable. The system will compute the travel time for the travel route by summing the travel time for the portion of each link that falls within the route (using the percentage of the link within the route times the total link travel time). If the computed travel time for the travel route exceeds the alert/notification travel time threshold, the system will create an alert and/or send a notification. The system will archive the calculated data for accountability purposes.

4.2.2.5 Create External Connection Alert (Use Case)

The system will create an External Connection Alert if an external connection transitions to the "failed" state and remains there for an amount of time, as specified in the connection settings, if the alert is enabled in the connection settings. The system will create an External Connection Alert if an external connection transitions to the "warning" state and remains there for an amount of time, as specified in the connection settings, if the alert is enabled in the connection settings. (The time spent in the "failed" state contributes to the time counted for a warning).

4.2.2.6 Create Travel Time Alert (Use Case)

If travel time alerts are enabled for a travel route, the system will issue an alert to the operations center specified in the travel route settings when the travel time crosses the threshold specified in the travel route settings.

4.2.2.7 Import INRIX Data (Use Case)

The system shall connect to the INRIX system and periodically import travel time data for links on a configurable interval. CHART will maintain the connection by periodically obtaining a new authentication key as required by the INRIX system. An External Connection Alert (when enabled for the INRIX connection) will be sent to the operations center configured to receive alerts for the INRIX connection. The following conditions will trigger an alert: Data cannot be retrieved from INRIX for a configurable period of time Data retrieved from INRIX does not comply with the documented format Data retrieved from INRIX does not contain data for a link that is included in a CHART travel route

4.2.2.8 Import INRIX Links (Use Case)

The system shall allow the administrator to import INRIX links from the INRIX distribution CD via an offline process. Running this import will completely replace any existing INRIX link data included in CHART with the new INRIX link data. Only links that fall within a system defined bounding rectangle will be imported. Note that the link data is read-only within the CHART system and is used to allow users to associate INRIX links with CHART travel routes (via the link IDs). The presence of these INRIX links within CHART is for the sole purpose of making that association process easier for the user.

4.2.2.9 Monitor External System Connection (Use Case)

The system shall monitor the state of the External System Connection.

4.2.2.10 Provide Link Data to Routes (Use Case)

The system shall provide link level data read from INRIX to an object or objects in the system that group link level data into routes.

4.2.2.11 Send External Connection Notification (Use Case)

The system will send a notification to a specified notification group if an external connection transitions to the "failed" state and remains there for an amount of time, as specified in the connection settings, if enabled in the connection settings. The system will send a notification to a specified notification group if an external connection transitions to the "warning" state and remains there for an amount of time, as specified in the connection settings, if enabled in the connection settings. (The time spent in the "failed" state contributes to the time counted for a warning).

4.2.2.12 Send Travel Time Notification (Use Case)

If travel time notifications are enabled for a travel route, the system will send a notification to the group specified in the travel route settings when the travel time crosses the threshold specified in the travel route settings.

4.2.2.13 System (Actor)

The System actor represents any software component of the CHART system. It is used to model uses of the system which are either initiated by the system on an interval basis, or are an indirect by-product of another use cases that another actor has initiated.

4.2.3 R3B3ImportVectorData (Use Case Diagram)

This diagram shows use cases related to importing toll rate data from the external Vector system.

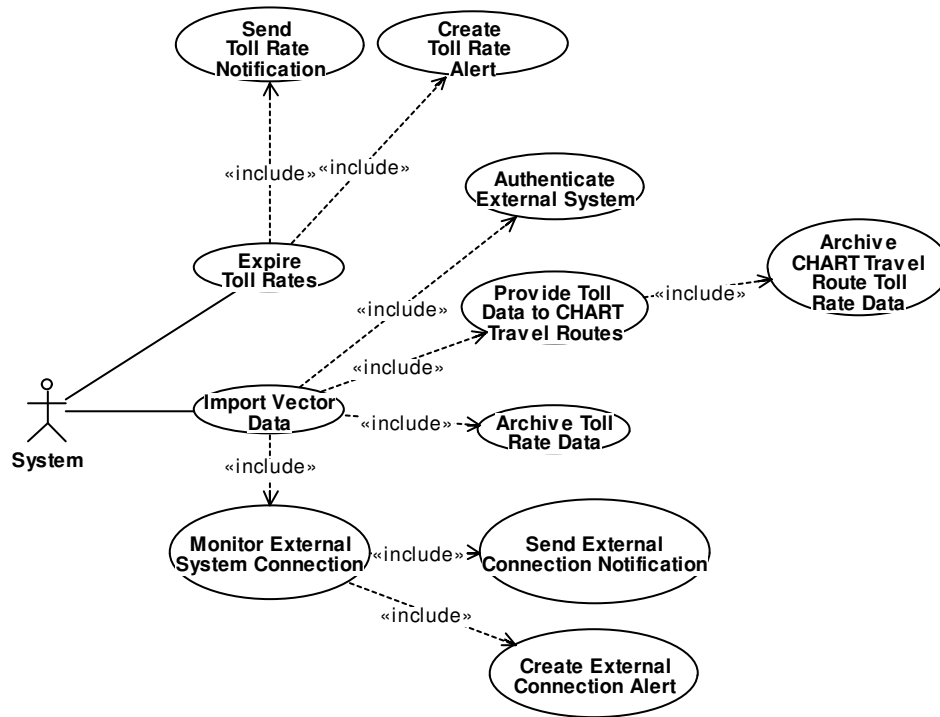


Figure 4-3 R3B3ImportVectorData (Use Case Diagram)

4.2.3.1 Archive CHART Travel Route Toll Rate Data (Use Case)

The system shall archive the toll rate data used by a travel route for a configurable period of time for the purpose of determining the composition of toll rate messages posted at a given time.

4.2.3.2 Archive Toll Rate Data (Use Case)

The system shall archive all toll rate data posted to the CHART system. The data shall be kept for offline inquiry for a configurable period of time.

4.2.3.3 Authenticate External System (Use Case)

The system shall authenticate external system connections to validate that they are authorized to connect to CHART and to enforce rights regarding the data the external system is permitted to access. Each external system owner will be provided a private key

from a public/private key pair generated within the CHART system. Each request from an external system will include the system's CHART client ID (as configured within CHART) and a digital signature of the request data, created using the private key provided by the CHART administrator. The CHART system will validate each request signature using the client's public key.

4.2.3.4 Create Toll Rate Alert (Use Case)

If toll rate alerts are enabled for a travel route, the system will issue an alert to the operations center specified in the travel route settings if the current toll rate for the travel route expires while there is a current (non-expired) toll rates document available.

4.2.3.5 Create External Connection Alert (Use Case)

The system will create an External Connection Alert if an external connection transitions to the "failed" state and remains there for an amount of time, as specified in the connection settings, if the alert is enabled in the connection settings. The system will create an External Connection Alert if an external connection transitions to the "warning" state and remains there for an amount of time, as specified in the connection settings, if the alert is enabled in the connection settings. (The time spent in the "failed" state contributes to the time counted for a warning).

4.2.3.6 Expire Toll Rates (Use Case)

When toll rate data is imported, it includes an optional expiration time for all toll rates included in the import. The system shall clear the toll rate data from all travel routes when the expiration time for the toll rate data used by a travel route passes. It is expected that the expiration time will be set such that it will only be used if Vector experiences a failure updating the toll rate data at its normal rate.

4.2.3.7 Import Vector Data (Use Case)

The system shall integrate with VECTOR for the purpose of asynchronously receiving toll rate information for routes configured within CHART. The system shall allow the VECTOR system to post toll rate update documents to a web service hosted at a configurable publicly accessible IP address and port. The system shall verify that posted data has originated from the VECTOR system by requiring that the posted data be digitally signed with a previously provided private key. The system shall validate all data posted by the VECTOR system against the published XSD. The system shall return a response XML document each time data is received from the VECTOR system. The system shall return a success code and a list of accepted toll routes each time the VECTOR system successfully posts a toll rate update. The system shall set the External Connection state to OK when a valid, complete message is received from the VECTOR system. The system shall return a failure code and a list of error code/error message pairs each time the VECTOR system posts invalid data. The system shall reject any toll rate data that does not have a digital signature with an authorizationError error code with corresponding error text. The system

shall reject any toll rate data that contains a digital signature that cannot be read with the previously provided public key by returning an `authorizationError` error code with corresponding error text. The system shall reject any toll rate data that does not validate correctly against the published XSD by returning an `invalidXML` error code with corresponding error text. The system shall reject any toll rate update that has a `startDateTime` that is more than a configurable number of minutes in the future by returning an `invalidStartDateTime` error code with corresponding error text. The system shall reject any toll rate update that has an `expirationDate` that is not later than the posted `startDateTime` by returning an `invalidExpirationDate` error code with corresponding error text. The system shall set the External Connection state to `WARNING` each time the VECTOR system posts data that fails validation or authorization. The system shall set the External Connection state to `WARNING` each time the VECTOR system posts data does not contain data for any toll route (start id/destination id pair) that is associated with a CHART system travel route. The system shall set the External Connection state to `FAILED` if it does not receive any data from the VECTOR system for a configurable period of time (expected to be on the order of 1 hour). The system shall reject in full any toll rate update from the VECTOR system that contains any errors and shall not use any toll rate from said update document. (Other than sending the appropriate error code in the XML response, the system shall otherwise react as if no document had been sent at all.) If configured to do so, the system shall generate an alert if it does not receive any data from the VECTOR system for a configurable period of time. If configured to do so, the system shall issue a notification if it does not receive any data from the VECTOR system for a configurable period of time. The system shall allow the VECTOR system to post toll rate update documents to a backup web service hosted at a configurable publicly accessible IP address and port. The system shall allow an administrator to configure alert and notification settings that cause an alert and/or notification to be issued when data is posted to the backup service. These settings shall include the ability to enable/disable these alerts and notifications, the op center to be alerted, the notification group to be notified, and the minimum interval at which notifications can be sent.

4.2.3.8 Monitor External System Connection (Use Case)

The system shall monitor the state of the External System Connection.

4.2.3.9 Provide Toll Data to CHART Travel Routes (Use Case)

The system shall provide toll rate data for each toll route provided by the Vector system to CHART travel routes configured to include the toll route. Any existing toll rate data for the travel route shall be replaced by the new toll rate data. If a toll rate import does not include the toll rate for a toll route that is configured to be included in a travel route, the toll rate data for that travel route shall be cleared. This includes the case where the toll route is not included in the data import and the case where the toll route is included but a toll rate for that toll route is not included.

4.2.3.10 Send Toll Rate Notification (Use Case)

If toll rate notifications are enabled for a travel route, the system will send a notification to the group specified in the travel route settings if the current toll rate for the travel route expires while there is a current (non-expired) toll rates document available.

4.2.3.11 Send External Connection Notification (Use Case)

The system will send a notification to a specified notification group if an external connection transitions to the "failed" state and remains there for an amount of time, as specified in the connection settings, if enabled in the connection settings. The system will send a notification to a specified notification group if an external connection transitions to the "warning" state and remains there for an amount of time, as specified in the connection settings, if enabled in the connection settings. (The time spent in the "failed" state contributes to the time counted for a warning).

4.2.3.12 System (Actor)

The System actor represents any software component of the CHART system. It is used to model uses of the system which are either initiated by the system on an interval basis, or are an indirect by-product of another use cases that another actor has initiated.

4.2.4 R3B3ManageTravelerInformationMessages (Use Case Diagram)

This diagram shows use cases related to configuring traveler information messages.

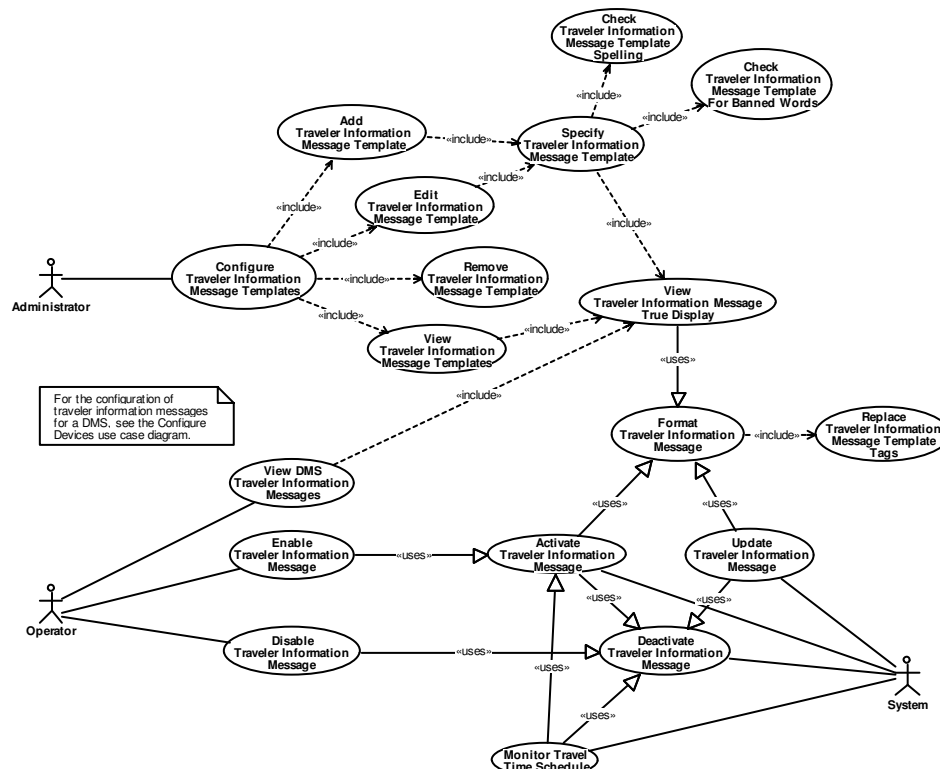


Figure 4-4 R3B3ManageTravelerInformationMessages (Use Case Diagram)

4.2.4.1 Activate Traveler Information Message (Use Case)

The system will activate a traveler information message on a DMS when the traveler information message has been enabled by the user, or if a currently enabled travel time message is currently not active due to the travel time display schedule and the schedule now indicates travel time messages may be displayed. The message will be activated on the device even if the device is not used by a traffic event. The system will format the message and place it on the DMS's arbitration queue. The level at which the message is added will be determined by whether a toll rate field is in the message template. If there is one, it will use the bin that specified for toll rate messages in the DMS configuration to determine which bin to use. (By default, it will go into the "Toll Rate" bin). If the template does not contain a toll rate field but does contain a travel time field, the message will be placed in the bin specified for travel time messages in the DMS configuration. (By default, it will go into the "Travel Time" bin). Once it is added to the queue, the message will follow priority and message combination rules to determine what message is displayed to drivers. The system will only allow one traveler information message to be active at a time, so it will deactivate

any previously active message before activating the new one. The system will add an entry to the operations log to record the message when a traveler information message is displayed on the DMS.

4.2.4.2 Add Traveler Information Message Template (Use Case)

A user with sufficient privileges will be able to add a new traveler information DMS message template. The user will specify a sign size for the template, and specify other aspects of the template as defined in the Specify Traveler Information Message Template use case.

4.2.4.3 Administrator (Actor)

An administrator is a CHART user that has functional rights assigned to allow them to perform administrative tasks, such as system configuration and maintenance.

4.2.4.4 Check Traveler Information Message Template For Banned Words (Use Case)

The system will check the traveler information message template for words that exist in the banned words dictionary and will not allow the message to be saved if it contains banned words.

4.2.4.5 Check Traveler Information Message Template Spelling (Use Case)

A user editing a traveler information message template will be able to perform a spell check on the message template.

4.2.4.6 Configure Traveler Information Message Templates (Use Case)

A user with sufficient rights will be able to configure the available DMS message templates that may be used for travel time / toll rate messages. The user will be able to add, edit, remove, and view these templates.

4.2.4.7 Deactivate Traveler Information Message (Use Case)

The system will support deactivating an active traveler information message, causing it to be removed from the arbitration queue. The message will be deactivated if the user disables the message, if another traveler information message is activated, if the currently active message or template is removed, or if the message becomes invalid or empty due to changes in the travel route data. A travel time message can be deactivated as specified in the travel time display schedule used by the DMS (the system-wide schedule or the schedule specified for the DMS). Note that only traveler information messages with at least one toll rate field are considered toll rate messages, not travel time messages, and therefore are unaffected by the travel time schedule even if they also contain travel time fields. The system will log a message to the operations log when a traveler information message is deactivated, whether it was initiated by the user or automatically.

4.2.4.8 Disable Traveler Information Message (Use Case)

The system shall allow a user with appropriate rights to disable a traveler information message that was previously enabled for a DMS. Disabling a message causes the system to deactivate it.

4.2.4.9 Edit Traveler Information Message Template (Use Case)

A user with sufficient privileges will be able to edit an existing traveler information DMS message template. The user cannot change the sign size for an existing template, but can change any other attribute as described in the Specify Traveler Information Message Template use case.

4.2.4.10 Enable Traveler Information Message (Use Case)

The system shall allow a user with appropriate rights to enable a traveler information message that has been configured for a DMS. Enabling a traveler information message allows it to be activated by the system.

4.2.4.11 Format Traveler Information Message (Use Case)

The system will format the traveler information message to obtain the final MULTI string to be sent to the DMS or displayed to the user. This formatting algorithm will use the settings specified in the traveler information message, which includes the message template and the travel routes. If available, the system will use current data from the travel route(s) used by the message. If a travel route cannot supply current data, the behavior will depend on a flag indicating whether to use dummy data for missing travel route data. The use of dummy data would apply if the message is being formatted for a message editor or simulation graphic. If it is not using dummy data (e.g., if it's building a message for use on the DMS), it will exclude the missing travel route data from the message and any associated message text, using the rules specified in the template (i.e., discard the row, page, or message). After the tags are replaced with data and any invalid portions of the message are discarded, the system will apply the automatic row formatting algorithm if it is requested in the message settings. If a page has one line of text, it will be placed on line two of a 3 or 4 line DMS. If a page has two lines of text, it will be placed on lines one and three of a 3 or 4 line DMS. If a page has three lines of text, it will be placed on the first three lines of a 3 or 4 line DMS.

4.2.4.12 Monitor Travel Time Schedule (Use Case)

If a message is a travel time message (and does not include toll rates) and is currently enabled, the system will use the travel time schedule (system-wide by default unless overridden in the DMS settings) to determine when to put the travel time message on the sign. If the active message is a travel time message, the system will use the travel time schedule to determine when to deactivate the message. It will activate or deactivate the message at the appropriate times.

4.2.4.13 Operator (Actor)

An operator is a user of the system who has been assigned a valid username/password combination and granted roles for system access.

4.2.4.14 Remove Traveler Information Message Template (Use Case)

A user with sufficient privileges will be able to remove a traveler information DMS message template from the system. The system will prompt the user for confirmation before removing the template. The system will prevent the template from being removed if it is used by a DMS.

4.2.4.15 Replace Traveler Information Message Template Tags (Use Case)

The system will replace tags in the traveler information message template when formatting the message. A destination tag will be replaced with the preferred, alternate 1, or alternate 2 destination name for a travel route will be used such that it fits within the allocated space for the tag. (Note that the destination tag can occupy all remaining space on a row, or it can be a fixed number of characters). The destination name will be justified within the space allocated for it using the justification specified in the message template. A travel time tag will be replaced with the actual travel time, using the format specified in the template. A travel time range tag will be replaced with a travel time range, according to the format specified in the template. The values for the range will be calculated by using the actual travel time and adding / subtracting the specified number of minutes as specified in the system profile. If a travel time falls below the minimum travel time specified for the travel route, the minimum value for the travel route will be used. The travel time / travel time range data will not be used if travel times are disabled for the travel route, the actual travel time exceeds the maximum travel time specified for the travel route, or the data does not meet the data quality threshold standard for a number of links exceeding the number specified for the travel route. A toll rate tag will be replaced with the toll rate, using the format specified in the message template. If toll rates are disabled for the travel route, the toll rate will not be used in the message. A toll rate time tag will be replaced with the latest toll rate time from any of the toll rate source (travel routes) used in the message for which toll rates are not disabled. The toll rate time will use the format specified in the message template. If toll rates are unavailable or disabled for all travel routes used for toll rates in the message, the toll rate time will not be used in the message. A route length tag will be replaced with the route length, using the format specified in the message template. For all tag types containing numbers, the numbers will be right justified within the space allocated for them within the tag, according to the specified format. If a travel route is not found, if it cannot provide the required data, or if the data is too large to fit in the allocated space, those tags will be ignored and the rules for missing data as specified in the message template will be applied.

4.2.4.16 Specify Traveler Information Message Template (Use Case)

A user adding or editing a traveler information DMS message template will be able to

specify a DMS message template for later use, which can be up to 2 pages long. The user will be able to edit the contents of each row of the message using a combination of freeform text and/or "tags" which act as placeholders for travel route information that will be filled in later. The user will be able to specify the line justification for each row (left, center, right). Tags types will include: destination name, travel time, travel time range, toll rate, toll rate time, and route length. These tags (with the exception of toll rate time) contain an index corresponding to a travel route to be assigned when the template is configured for a specific DMS. The use of indexes will allow data for multiple routes to be represented in the same template, and up to 6 routes may be used. By default the "destination" tag occupies all remaining space on a row of the message (negating any line justification for that row), but the user can also restrict a destination tag to a specified size. The user will be able to specify the justification within all destination tags (left/right/center) in case there is extra space. The other tags - travel time, travel time range, toll rate, toll rate time, and route length - all contain numbers. Within the numeric portion(s) of the tags the numbers will be right justified. For each of the numeric tag types that are used in the message, the user will be able to select one of several different number formats of various lengths, which may be necessary to get the message to fit on smaller signs. The user can specify the behavior to follow if an assigned travel route cannot provide the data at run time, including: discarding the row, discarding the page, or discarding the entire message. The user can also specify the page on / page off times for the message, and the description for the template. If the text of the message becomes too long for a row when the user is editing, the system will display a warning message.

4.2.4.17 System (Actor)

The System actor represents any software component of the CHART system. It is used to model uses of the system which are either initiated by the system on an interval basis, or are an indirect by-product of another use cases that another actor has initiated.

4.2.4.18 Update Traveler Information Message (Use Case)

The system will update an active traveler information message when new data is received for the travel routes in the message. The system will update the message if the travel route settings are modified (for example, if travel times and/or toll rates are enabled or disabled). The system will also update the message if the message template is changed (e.g., by the user), or if the message is changed as it is configured for a DMS. The system will reformat the message and will update the message on the DMS if the message has changed. The message will remain on the arbitration queue during this process and will not lose its place on the queue unless the new data causes the message to become invalid or empty, in which case it will be automatically deactivated. The system will add an entry to the operations log to record the message when a traveler information message is updated on the DMS.

4.2.4.19 View DMS Traveler Information Messages (Use Case)

The system will allow the user to view the traveler information messages configured for a DMS. It will display a graphical representation of the message, using actual data from the

routes if available, or dummy data if not available. It will also display the enabled / disabled status of the messages.

4.2.4.20 View Traveler Information Message Templates (Use Case)

A user with sufficient rights will be able to view the list of traveler information DMS message templates. The system will display the template name, sign size, textual and graphical representations of the message. It will also display a representation of the formats used in the template for travel time, travel time range, toll rate, toll rate time, and route length. The list will be sortable on template name, sign size, travel time format, travel time range format, toll rate format, toll rate time format, and route length format. The list will be filterable on sign size, travel time format, travel time range format, toll rate format, toll rate time format, and route length format. The filterable values for the formats will include values indicating whether or not the template contains a tag of the corresponding type.

4.2.4.21 View Traveler Information Message True Display (Use Case)

The system shall allow the user to preview the traveler information message as a graphical representation. The message will be formatted for display as described in the Format Traveler Information Message use case. The graphic will be updated if the message changes (although if the message exceeds the sign size, the image may not be updated).

4.2.5 R3B3ManageDeviceQueue (Use Case Diagram)

This diagram shows uses of the system related to managing device message queues. For R3B3, the existing message queue functionality is being enhanced to support traveler information messages. Two new queue priorities are being added for travel time and toll rate messages. Traveler information messages will be added to the DMS device queue when activated by the system, and will be removed from the DMS device queue when deactivated by the system.

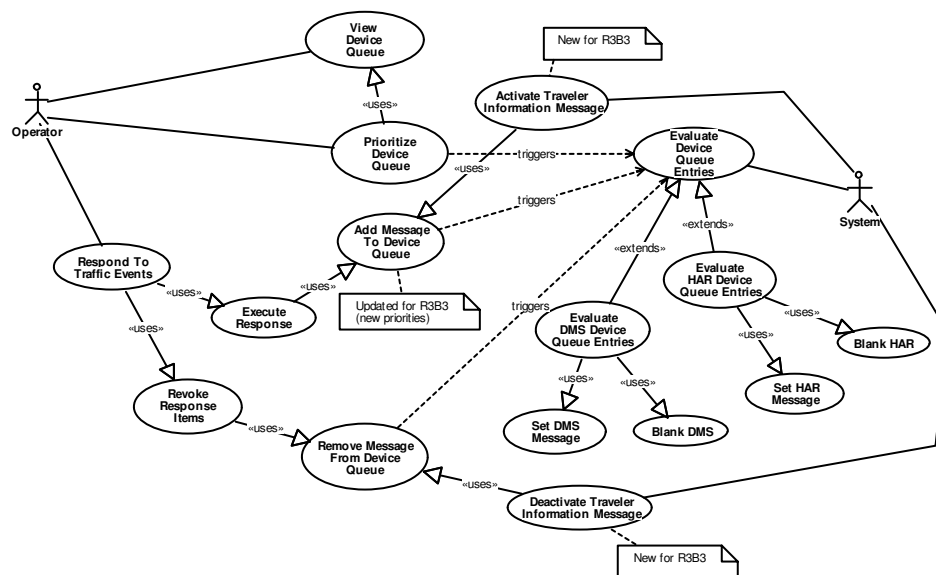


Figure 4-5 R3B3ManageDeviceQueue (Use Case Diagram)

4.2.5.1 Activate Traveler Information Message (Use Case)

The system will activate a traveler information message on a DMS when the traveler information message has been enabled by the user, or if a currently enabled travel time message is currently not active due to the travel time display schedule and the schedule now indicates travel time messages may be displayed. The message will be activated on the device even if the device is not used by a traffic event. The system will format the message and place it on the DMS's arbitration queue. The level at which the message is added will be determined by whether a toll rate field is in the message template. If there is one, it will use the bin that specified for toll rate messages in the DMS configuration to determine which bin to use. (By default, it will go into the "Toll Rate" bin). If the template does not contain a toll rate field but does contain a travel time field, the message will be placed in the bin specified for travel time messages in the DMS configuration. (By default, it will go into the "Travel Time" bin). Once it is added to the queue, the message will follow priority and message combination rules to determine what message is displayed to drivers. The system will only allow one traveler information message to be active at a time, so it will deactivate

any previously active message before activating the new one. The system will add an entry to the operations log to record the message when a traveler information message is displayed on the DMS.

4.2.5.2 Add Message To Device Queue (Use Case)

When a traffic event response plan is executed, the specified messages are placed in the specified devices' arbitration queues. A message may also be placed in a device's arbitration queue when a traveler information message is activated on a DMS. When an item is added to an arbitration queue, it is held with other messages that have been added to the queue. The system selects the highest priority message to display / play on the device. A message that does not have the highest priority at one time may become the message with highest priority in the future due to messages being removed from the queue or by manual intervention by an operator. For R3B3, two new message queue priorities are being added to the arbitration queue for traveler information messages. The "Toll Rate" priority (7) is the default priority for traveler information messages that contain at least one toll rate field. The "Travel Time" priority (6) is the default priority for traveler information messages that contain at least one travel time field but no toll rate fields. These default priorities can be overridden for each DMS, causing traveler information messages to be added to the queue with a priority as specified by an administrator rather than these defaults. The system stores the event a message is associated with and the operations center responsible for the message if the message is activated via a traffic event response. These fields do not apply to traveler information messages.

4.2.5.3 Blank DMS (Use Case)

A DMS can be blanked when the DMS is online or in maintenance mode. When the DMS is online, it is only blanked by the device's arbitration queue when the arbitration queue becomes empty. When the DMS is in maintenance mode, the DMS can be blanked directly by the user if they have the proper functional rights.

A DMS can be blanked indirectly by other commands, such as placing the device online, offline or in maintenance mode or by resetting the device.

When a DMS that has beacons is blanked, its beacons are turned off.

4.2.5.4 Blank HAR (Use Case)

A HAR can be blanked if it is online or in maintenance mode. When the HAR is online, the device is only blanked if there are no traffic events that have currently requested that a message be placed on the device. When the HAR is in maintenance mode, the HAR can be blanked directly by the user.

A HAR can be blanked indirectly through administrative functions such as placing the device online or resetting the device.

When a HAR is blanked, the system will set the HAR's default message to be the current

message. Additionally, the system will deactivate any associated active SHAZAMs before blanking the HAR itself.

4.2.5.5 Deactivate Traveler Information Message (Use Case)

The system will support deactivating an active traveler information message, causing it to be removed from the arbitration queue. The message will be deactivated if the user disables the message, if another traveler information message is activated, if the currently active message or template is removed, or if the message becomes invalid or empty due to changes in the travel route data. A travel time message can be deactivated as specified in the travel time display schedule used by the DMS (the system-wide schedule or the schedule specified for the DMS). Note that only traveler information messages with at least one toll rate field are considered toll rate messages, not travel time messages, and therefore are unaffected by the travel time schedule even if they also contain travel time fields. The system will log a message to the operations log when a traveler information message is deactivated, whether it was initiated by the user or automatically.

4.2.5.6 Evaluate Device Queue Entries (Use Case)

When the contents of a device queue are altered, it shall evaluate the entries on the queue to determine what action (if any) to take on its associated device. When a device queue becomes empty, it shall blank its corresponding device. When a device queue is evaluated and it is not empty, it shall choose the highest priority message and set this message as the device's current message if it is not set already.

4.2.5.7 Evaluate DMS Device Queue Entries (Use Case)

The system shall evaluate entries placed on a DMS's arbitration queue in response to traffic events. The system shall use a priority algorithm to determine which message shall be placed on the DMS device. The system shall evaluate entries when a new entry is added, when an entry is removed, and when notified by the DMS device object that a previous asynchronous request has completed. When the queue is evaluated, the highest priority message shall be set on the DMS device, unless it is currently already set on the DMS device. When an evaluation occurs and the queue has become empty, the queue shall blank the DMS. The queue shall allow the concatenation of 2 single page messages to be set on the DMS device according to certain rules and configuration settings. The rules that govern this message concatenation feature are specified in the system profile.

4.2.5.8 Evaluate HAR Device Queue Entries (Use Case)

The system shall evaluate entries placed on a HAR's arbitration queue in response to traffic events. The system shall use a priority algorithm to determine which message shall be placed on the HAR device. The system shall evaluate entries when a new entry is added, when an entry is removed, and when notified by the HAR device object that a previous asynchronous request has completed. When the queue is evaluated, the highest priority message shall be set on the HAR device, unless it is already currently set on the HAR

device. When an evaluation occurs and the queue has become empty, the queue shall set the HAR device to its default message. The queue shall allow the concatenation of multiple messages to be set on the HAR device as the recording space on the HAR allows and according to configuration settings and concatenation rules. The rules that govern this feature are specified by an administrator in the system profile.

4.2.5.9 Execute Response (Use Case)

An operator with the correct functional rights may execute the response plan for a particular traffic event. Performing this operation will place the message from each response plan item on the arbitration queue of the corresponding device.

4.2.5.10 Operator (Actor)

An operator is a user of the system who has been assigned a valid username/password combination and granted roles for system access.

4.2.5.11 Prioritize Device Queue (Use Case)

A user with the proper functional rights can manually change the priority of items on a device's queue to override the queue's automated prioritization scheme. If this manual re-prioritization causes a message on the queue to have a priority higher than the message that is currently on the device, the device's queue will change the message to the message of highest priority.

4.2.5.12 Remove Message From Device Queue (Use Case)

When a response plan item is removed from a traffic event's response plan, the item removes its message from the queue of the device specified in the item. This causes the queue to evaluate the remaining messages on the queue (if any) and either set the device to the next highest priority message or blank the device.

4.2.5.13 Respond To Traffic Events (Use Case)

A user with proper functional rights can create a response plan associated with a traffic event. This response plan defines DMS and HAR devices to be used to help manage the traffic event along with the message to be placed on each device. After setting up or changing entries in a response plan, the user can execute the plan. The user can also execute individual items in the plan. When a user no longer wishes to use a specific device in the response to the traffic event, the user may remove the item from the response plan. When the user closes the traffic event, all items used in the traffic event response plan are automatically removed. The inclusion of a device in a response plan is a request by the user for the device to display the message. The message is only displayed if there are no traffic events of higher priority that have also requested that a message be displayed on the device.

4.2.5.14 Revoke Response Items (Use Case)

A user with the proper functional rights can remove a device from the response plan of a traffic event. The system will also automatically perform this operation when a traffic event is closed. When a response plan item is removed from the response plan, the message specified in the item is removed from the specified device's arbitration queue.

4.2.5.15 Set DMS Message (Use Case)

The message on a DMS can be set when the DMS is online or in maintenance mode. When the DMS is online, the message is set by the DMS's arbitration queue. This queue sets the message of the DMS to be the message that is on the queue that has the highest priority. When the DMS is in maintenance mode, an operator with proper functional rights can set the message on a DMS directly.

4.2.5.16 Set HAR Message (Use Case)

A HAR's message is set through the execution of an event response plan or set directly by an administrator when the device is in maintenance mode. The message activation may specify messages which were previously stored in message slots in the controller or a message that was created using the HAR message editor.

When activating a HAR message created by the message editor the user may choose to use the default header and trailer or just use the message body for the entire message. Messages activated in this manner shall be loaded into the HAR controller in the slot designated for immediate broadcast.

A HAR message activation also specifies if each associated SHAZAM should be activated or not.

4.2.5.17 System (Actor)

The System actor represents any software component of the CHART system. It is used to model uses of the system which are either initiated by the system on an interval basis, or are an indirect by-product of another use cases that another actor has initiated.

4.2.5.18 View Device Queue (Use Case)

An operator with proper functional rights can view the messages that are queued for a device. The user shall be able to see the actual message to be set or a description of the message (in the case of a voice message) and the current priority of each message.

4.2.6 R3B3ConfigureDMSTravelerInfoMsgSettings (Use Case Diagram)

This diagram shows use cases related to configuring the DMS settings related to traveler information messages.

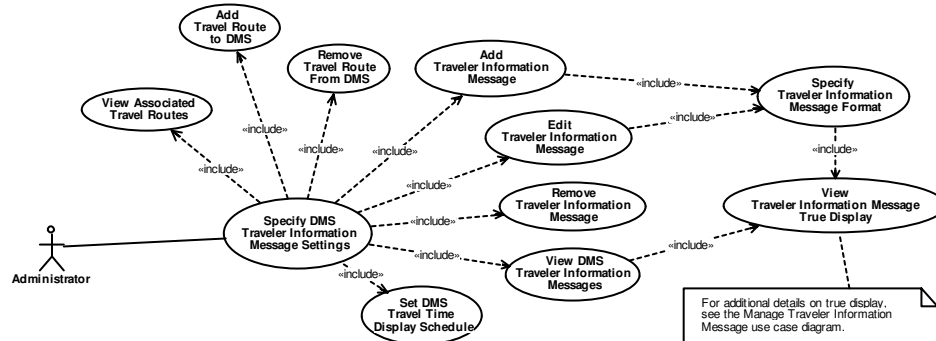


Figure 4-6 R3B3ConfigureDMSTravelerInfoMsgSettings (Use Case Diagram)

4.2.6.1 Add Travel Route to DMS (Use Case)

The user shall be able to add a travel route to a DMS so that it is available for selection when configuring traveler information messages for the DMS.

4.2.6.2 Add Traveler Information Message (Use Case)

A user with sufficient rights will be able to add a traveler information message to a DMS, so that the message will be available for later use by an operator. A newly added message will be disabled by default.

4.2.6.3 Administrator (Actor)

An administrator is a CHART user that has functional rights assigned to allow them to perform administrative tasks, such as system configuration and maintenance.

4.2.6.4 Edit Traveler Information Message (Use Case)

A user with sufficient rights will be able to edit a traveler information message that was previously added to a DMS. The user will be able to change anything specified when adding the message to the DMS (which does not include editing the message template itself). See the Specify Traveler Information Message Format use case for details.

4.2.6.5 Remove Travel Route From DMS (Use Case)

The user shall be able to remove a travel route from a DMS so that it is no longer available for selection when configuring traveler information messages for the DMS.

4.2.6.6 Remove Traveler Information Message (Use Case)

A user with sufficient rights will be able to remove a traveler information message that was previously added to a DMS. The system will ask the user for confirmation, and will automatically disable the message before removing it.

4.2.6.7 Set DMS Travel Time Display Schedule (Use Case)

The system shall allow a user with appropriate rights to set the travel time display schedule for a DMS. The user can choose to use the system-wide schedule or to override it for the DMS. The schedule shall include one or more time ranges when travel times should be displayed, or it may be a 24/7 schedule.

4.2.6.8 Specify Traveler Information Message Format (Use Case)

The system shall allow a user with appropriate rights to set the format used to display a traveler information message for a DMS. The user will be able to select one of the previously defined message templates that exactly match the sign size. The system will display a textual representation of the message template including the tags, which contain index values indicating which travel route selection will serve as the data source for the tag. For each travel route index used in the message, the user will be able to select from among the travel routes associated with the DMS that can provide the data to satisfy all tags having the given index. The default travel route of "None" will always be available, and this will be the only option if there are no associated travel routes that can supply the data requested by the tag. When specifying the message, a graphical representation of the message will be displayed, and it will use actual data from the source if available, or dummy data if not available. (See the View Traveler Information Message True Display use case for details). (If routes are not selected (i.e., "None" is selected), the graphic will degrade according to the rules specified in the template.) The user will be able to specify whether the rows of the message (after the message is degraded due to missing data, if any) will be aligned automatically.

4.2.6.9 Specify DMS Traveler Information Message Settings (Use Case)

A user with appropriate rights shall be permitted to configure a DMS for displaying traveler information messages for travel times and/or toll rate information. The user will be able to associate travel routes with the DMS to make them available for use by the message templates, and the user will be able to view the currently associated travel routes. The user will be able to manage (create, edit, delete, and view) traveler information messages for a DMS. Traveler information messages use pre-defined message templates and associated travel routes. The traveler information messages are configured in advance for a DMS so that an operator can use the messages without having to edit them. The user will be able to specify the arbitration queue levels at which travel time and toll rate messages will be added. (By default, these levels will be the Travel Time and Toll Rate levels, but other levels may be selected). The user will be able to specify a travel time display schedule for a DMS that overrides the system-wide display schedule. See the specific use cases for details.

4.2.6.10 View Associated Travel Routes (Use Case)

The system shall allow the user to view the travel routes associated with the DMS.

4.2.6.11 View DMS Traveler Information Messages (Use Case)

The system will allow the user to view the traveler information messages configured for a DMS. It will display a graphical representation of the message, using actual data from the routes if available, or dummy data if not available. It will also display the enabled / disabled status of the messages.

4.2.6.12 View Traveler Information Message True Display (Use Case)

The system shall allow the user to preview the traveler information message as a graphical representation. The message will be formatted for display as described in the Format Traveler Information Message use case. The graphic will be updated if the message changes (although if the message exceeds the sign size, the image may not be updated).

4.3 Device Enhancements

4.3.1 R3B3ConfigureDevices (Use Case Diagram)

This use case diagram shows use cases related to device configurations.

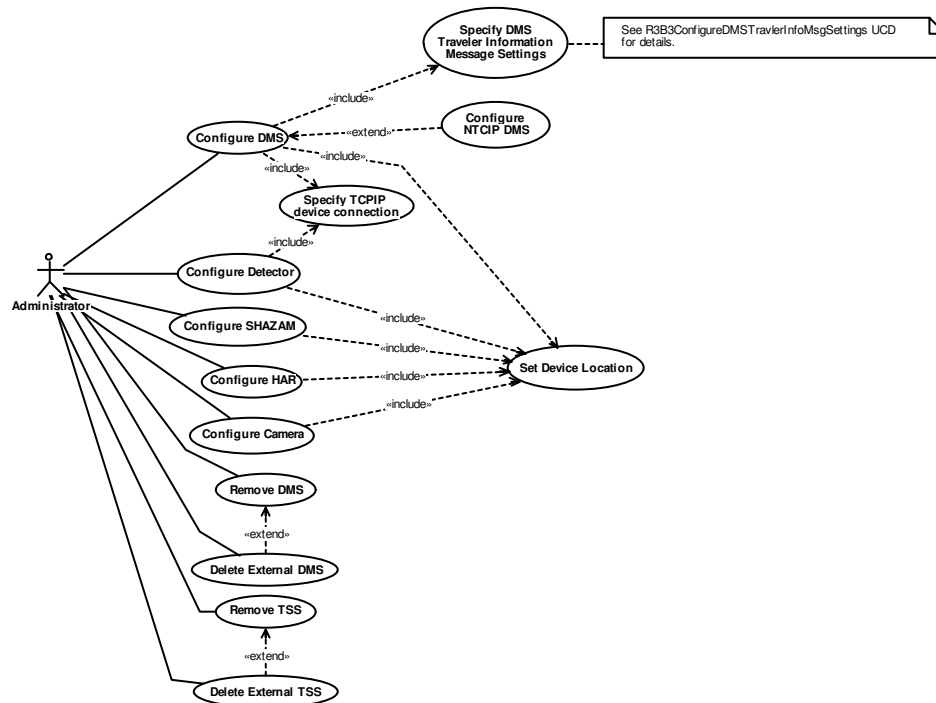


Figure 4-7 R3B3ConfigureDevices (Use Case Diagram)

4.3.1.1 Administrator (Actor)

An administrator is a CHART user that has functional rights assigned to allow them to perform administrative tasks, such as system configuration and maintenance.

4.3.1.2 Configure NTCIP DMS (Use Case)

The system shall allow a user with appropriate rights to configure an NTCIP DMS. This feature exists in R3B2 and is being enhanced in R3B3 to include the setting of the default font and line spacing.

4.3.1.3 Configure Camera (Use Case)

The system shall allow a user with appropriate rights to configure cameras. This feature exists in R3B2 and is being enhanced in R3B3 to include more location information.

4.3.1.4 Configure Detector (Use Case)

The system shall allow a user with appropriate rights to configure the settings for a detector (TSS), unless it is an external device. This feature exists in R3B2 and is being enhanced in R3B3 to include more location information and the ability to use TCP/IP to communicate with the device.

4.3.1.5 Configure DMS (Use Case)

The system shall allow a user with appropriate rights to configure the settings for a DMS, unless it is an external DMS. This feature exists in R3B2 and is being enhanced in R3B3 to include more location information, configuration of traveler information message settings and the ability to use TCP/IP for communication. For NTCIP DMSs, settings are being added for default font and line spacing. Settings will include the operations center to send an alert to when the DMS goes into hardware failure, the operations center to send an alert to when the DMS goes into communications failure, the notification group to send a notification to when the DMS goes into hardware failure, and the notification group to send a notification to when the DMS goes into communication failure.

4.3.1.6 Configure HAR (Use Case)

The system shall allow a user with appropriate rights to configure HAR devices. This feature exists in R3B2 and is being enhanced in R3B3 to include more location information.

4.3.1.7 Configure SHAZAM (Use Case)

The system shall allow a user with appropriate rights to configure SHAZAM devices. This feature exists in R3B2 and is being enhanced in R3B3 to include more location information.

4.3.1.8 Delete External DMS (Use Case)

The system shall allow a suitably privileged user to delete an external DMS from the CHART system. When doing so, the DMS shall remain as a candidate DMS that is marked as being excluded from the CHART system. The administrator can choose to include the DMS at a later time.

4.3.1.9 Delete External TSS (Use Case)

The system shall allow a suitably privileged user to delete an external TSS from the CHART system. When doing so, the TSS shall remain as a candidate TSS that is marked as being excluded from the CHART system. The administrator can choose to include the TSS at a later time.

4.3.1.10 Remove DMS (Use Case)

A user with appropriate rights can remove a DMS from the CHART system.

4.3.1.11 Remove TSS (Use Case)

A user with appropriate rights can remove a TSS from the CHART system.

4.3.1.12 Set Device Location (Use Case)

The system shall allow a user with appropriate rights to set the location information for a device, including: location description, latitude/longitude, state, county, route type, route number (or name, and flag to indicate which to use), direction (including bidirectional directions), proximity to intersecting feature (intersection or state milepost), and intersecting feature information. The intersecting feature information will include state milepost number or intersecting route type, route number (or name, and flag to indicate which to use). The system will validate the geographical coordinates (if entered) against system-wide bounds defined in the system profile to make sure that they are not unreasonable. The location description is a required field, and it will be automatically generated by the system but may be overridden by the user. The system will prompt for confirmation when the user attempts to override the generated location description.

4.3.1.13 Specify DMS Traveler Information Message Settings (Use Case)

A user with appropriate rights shall be permitted to configure a DMS for displaying traveler information messages for travel times and/or toll rate information. The user will be able to associate travel routes with the DMS to make them available for use by the message templates, and the user will be able to view the currently associated travel routes. The user will be able to manage (create, edit, delete, and view) traveler information messages for a DMS. Traveler information messages use pre-defined message templates and associated travel routes. The traveler information messages are configured in advance for a DMS so that an operator can use the messages without having to edit them. The user will be able to specify the arbitration queue levels at which travel time and toll rate messages will be added. (By default, these levels will be the Travel Time and Toll Rate levels, but other levels may be selected). The user will be able to specify a travel time display schedule for a DMS that overrides the system-wide display schedule. See the specific use cases for details.

4.3.1.14 Specify TCPIP device connection (Use Case)

The system shall allow a user with appropriate rights to specify TCP/IP connection information for a device, including the IP address and port for the device.

4.3.2 R3B3ViewDeviceLists (Use Case Diagram)

This diagram shows use cases related to the operator viewing device lists. In R3B3, the operator can configure the specific columns displayed.

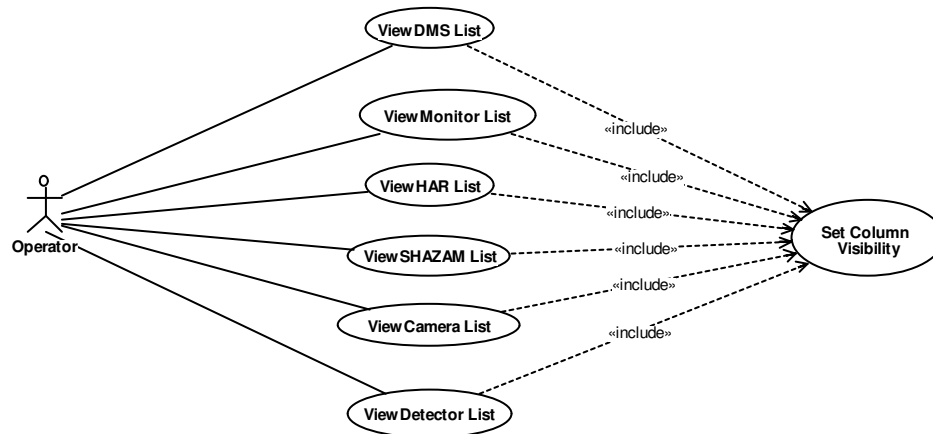


Figure 4-8 R3B3ViewDeviceLists (Use Case Diagram)

4.3.2.1 Operator (Actor)

An operator is a user of the system who has been assigned a valid username/password combination and granted roles for system access.

4.3.2.2 Set Column Visibility (Use Case)

The user can choose to show or hide columns that appear within a list. The user cannot hide the column that contains the object's name. The system will persist the user's choices for columns to be displayed per list type, and will display the same columns the next time the list is displayed. Preferences will be stored as cookies on the user's machine.

4.3.2.3 View Camera List (Use Case)

The system will allow a user with appropriate rights to view the list of cameras defined in the system. This feature exists in R3B2 and is enhanced for R3B3. New columns will include the route, county, direction, milepost, owning organization, and connection site. To save screen space, the visible columns will be selectable. Several of the new columns (milepost, owning organization, and connection site) will be hidden by default to save space. The user will be able to sort the camera list by county, route, direction, connection site, state milepost, and owning organization name (in addition to the columns that already support sorting in R3B2). The user will be able to filter the camera list by county, route, direction, connection site, and owning organization name (in addition to the columns that already support sorting in R3B2).

4.3.2.4 View Detector List (Use Case)

The system will allow a user with appropriate rights to view the list of detectors (TSSs) defined in the system. This feature exists in R3B2 and is enhanced for R3B3 to show new columns and to show either the actual or summary speeds for a detector based on the user's rights and the owning organization of the detector. New columns will include the route, county, direction, milepost, owning organization, port managers, and connection site. To save screen space, the visible columns will be selectable. Several of the new columns (milepost, owning organization, port managers, and connection site) will be hidden by default to save space. The user will be able to sort the TSS list by county, route, direction, port manager names, connection site, state milepost, and owning organization name (in addition to the columns that already support sorting in R3B2). The user will be able to filter the TSS list by county, route, direction, port manager names, connection site, and owning organization name (in addition to the columns that already support sorting in R3B2). The system will allow the user to filter the list to include or exclude external TSSs and/or internal (CHART) TSSs if the user has rights to view external TSSs; otherwise, external TSSs will be filtered out. If external TSSs are displayed, the user will be able to filter the list by agency.

4.3.2.5 View DMS List (Use Case)

The system will allow a user with appropriate rights to view the list of DMSs defined in the system. This feature exists in R3B2 and is enhanced for R3B3. New columns will include the route, county, direction, milepost, owning organization, port managers, connection site, and travel time schedule overridden indicator. To save screen space, the visible columns will be selectable. Several of the new columns (milepost, owning organization, port managers, connection site, and travel time schedule overridden indicator) will be hidden by default to save space. The user will be able to sort the DMS list by county, route, direction, port manager names, connection site, travel time schedule overridden indicator, state milepost, and owning organization name (in addition to the columns that already support sorting in R3B2). The user will be able to filter the DMS list by county, route, direction, port manager names, connection site, travel time schedule overridden indicator, and owning organization name (in addition to the columns that already support sorting in R3B2). The system will allow the user to filter the list to include or exclude external DMSs and/or internal (CHART) DMSs if the user has rights to view external DMSs; otherwise, external DMSs will be filtered out. If external DMSs are displayed, the user will be able to filter the list by agency.

4.3.2.6 View HAR List (Use Case)

The system will allow a user with appropriate rights to view the list of HARs defined in the system. This feature exists in R3B2 and is enhanced for R3B3. New columns will include the route, county, direction, milepost, owning organization, port managers, and connection site. To save screen space, the visible columns will be selectable. Several of the new columns (milepost, owning organization, port managers, and connection site) will be hidden by default to save space. The user will be able to sort the HAR list by county, route,

direction, port manager names, connection site, state milepost, and owning organization name (in addition to the columns that already support sorting in R3B2). The user will be able to filter the HAR list by county, route, direction, port manager names, connection site, and owning organization name (in addition to the columns that already support sorting in R3B2).

4.3.2.7 View Monitor List (Use Case)

The system shall allow a user with the proper rights to view a list of monitors defined in the system. This feature exists in R3B2 and is enhanced in R3B3. A network connection site column will be added, which will be hidden by default, but (when visible) will allow the user to sort and filter the list by connection site name. To save screen space, the visible columns will be selectable.

4.3.2.8 View SHAZAM List (Use Case)

The system will allow a user with appropriate rights to view the list of SHAZAMs defined in the system. This feature exists in R3B2 and is enhanced for R3B3. New columns will include the route, county, direction, milepost, owning organization, port managers, and connection site. To save screen space, the visible columns will be selectable. Several of the new columns (milepost, owning organization, port managers, and connection site) will be hidden by default to save space. The user will be able to sort the SHAZAM list by county, route, direction, port manager names, connection site, state milepost, and owning organization name (in addition to the columns that already support sorting in R3B2). The user will be able to filter the SHAZAM list by county, route, direction, port manager names, connection site, and owning organization name (in addition to the columns that already support sorting in R3B2).

4.3.3 R3B3ViewDeviceDetails (Use Case Diagram)

This diagram illustrates the functionality to display detailed device information. For R3B3, device location information will be enhanced for most device types. DMS and TSS details include TCP/IP configuration data. DMS details include traveler information messages and NTCIP message settings. The system enforces rights to restrict access to sensitive device configuration information, external DMS's/TSS's, and detailed lane specific data for TSS's.

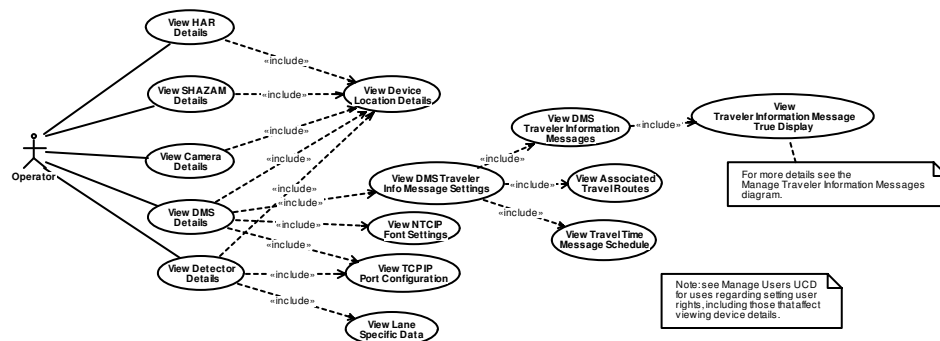


Figure 4-9 R3B3ViewDeviceDetails (Use Case Diagram)

4.3.3.1 Operator (Actor)

An operator is a user of the system who has been assigned a valid username/password combination and granted roles for system access.

4.3.3.2 View Associated Travel Routes (Use Case)

The system shall allow the user to view the travel routes associated with the DMS.

4.3.3.3 View Camera Details (Use Case)

The system shall allow a user with appropriate rights to view the details of a camera. This feature exists in R3B2 and is enhanced in R3B3 to show additional location information.

4.3.3.4 View Detector Details (Use Case)

The system shall allow a user with appropriate rights to view the details of a detector (TSS). This feature exists in R3B2 and is enhanced in R3B3 to show additional fields and to provide a read-only version for external detectors. New user rights are also added in R3B3 to allow more granular control of the data displayed on the page based on the device's organization and the rights granted to the user. Two new user rights will determine the type of Volume, Speed, and Occupancy (VSO) data the user will see - either the actual VSO data, just a speed range, or no VSO data. Another new user right will determine if the user can view sensitive configuration data for a TSS.

4.3.3.5 View Device Location Details (Use Case)

The system shall allow a user with appropriate rights to view the location information for a device.

4.3.3.6 View DMS Details (Use Case)

The system shall allow a user with appropriate rights to view the details of a DMS device. This feature exists in R3B2 and is enhanced in R3B3 to show additional fields, and to provide a read-only version for external DMSs.

4.3.3.7 View DMS Traveler Info Message Settings (Use Case)

The system shall allow a user with proper rights to view the DMS settings related to traveler information messages.

4.3.3.8 View DMS Traveler Information Messages (Use Case)

The system will allow the user to view the traveler information messages configured for a DMS. It will display a graphical representation of the message, using actual data from the routes if available, or dummy data if not available. It will also display the enabled / disabled status of the messages.

4.3.3.9 View HAR Details (Use Case)

The system shall allow a user with appropriate rights to view the details of a HAR device. This feature exists in R3B2 and is enhanced in R3B3 to show location information.

4.3.3.10 View Lane Specific Data (Use Case)

The system shall allow a user with appropriate rights to view the lane specific data for a detector, as provided by the detector. For an RTMS detector, this will likely be a lane number (0-7) and associated traffic parameters (speed, vol, occ).

4.3.3.11 View NTCIP Font Settings (Use Case)

The system shall display the font related settings for NTCIP model DMSs. This includes the default font number and default line spacing.

4.3.3.12 View SHAZAM Details (Use Case)

The system shall allow a user with appropriate rights to view the details of a SHAZAM device. This feature exists in R3B2 and is enhanced in R3B3 to show additional location information.

4.3.3.13 View TCP IP Port Configuration (Use Case)

The system shall allow a user with proper rights to view the TCP/IP connection

configuration for a device that supports such connections.

4.3.3.14 View Travel Time Message Schedule (Use Case)

The system shall allow the user to view the travel time display schedule for the DMS. The schedule specifies when travel times may be displayed on the DMS. Whether or not travel times are actually displayed depend on whether or not a traveler information message is currently active and whether or not it includes travel times.

4.3.3.15 View Traveler Information Message True Display (Use Case)

The system shall allow the user to preview the traveler information message as a graphical representation. The message will be formatted for display as described in the Format Traveler Information Message use case. The graphic will be updated if the message changes (although if the message exceeds the sign size, the image may not be updated).

4.3.4 R3B3ManageTrafficEvents (Use Case Diagram)

This diagram shows use cases related to managing traffic events. In R3B3, the operator can find close devices and add them to an event's response plan. R3B3 introduces the ability to transfer traffic event responsibility between Op Centers. New rights for controlling access to traffic event details data are included in R3B3. In R3B3 associations are automatically made between CHART and external events. In addition, new columns are added to the traffic event list.

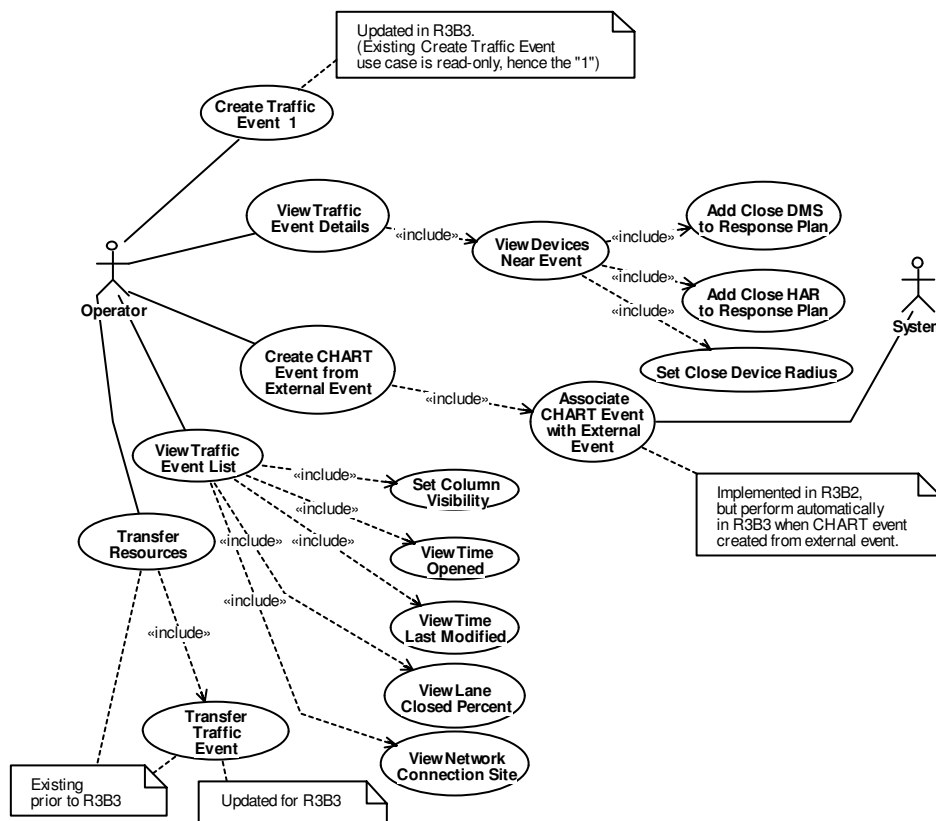


Figure 4-10 R3B3ManageTrafficEvents (Use Case Diagram)

4.3.4.1 Add Close DMS to Response Plan (Use Case)

The system allows appropriately privileged users to add a DMS listed as "close" to a traffic event to the event's response plan. The DMS will be added without a message. This feature only applies to CHART DMSs (External DMSs are excluded).

4.3.4.2 Add Close HAR to Response Plan (Use Case)

The system allows appropriately privileged users to add a HAR listed as "close" to a traffic event to the event's response plan. The HAR will be added with the HAR's default (empty)

message.

4.3.4.3 Associate CHART Event with External Event (Use Case)

The system shall allow a CHART event to be associated with an event imported from an external system. This feature exists in R3B2 and will be enhanced in R3B3 to perform this association automatically if an external event is used to create a new opened CHART event.

4.3.4.4 Create CHART Event from External Event (Use Case)

The system shall allow a user with proper rights to create a new, opened CHART event using the data from an existing external event. The system shall automatically create an association between the new event and the external event.

4.3.4.5 Create Traffic Event 1 (Use Case)

The user can create a new traffic event. This functionality is being updated in R3B3 to assign an owning organization to the traffic event when it is created. The owning organization assigned will be the owning organization of the user's operations center.

4.3.4.6 Operator (Actor)

An operator is a user of the system who has been assigned a valid username/password combination and granted roles for system access.

4.3.4.7 Set Close Device Radius (Use Case)

The user can select the radius used to define how close a device must be to the event for the device to be shown in the "close devices" section of an event details page. The user's radius setting for an event will stay in effect until the user changes the setting or they log out, even if they navigate away from the event details page and return to it at a later time.

4.3.4.8 Set Column Visibility (Use Case)

The user can choose to show or hide columns that appear within a list. The user cannot hide the column that contains the object's name. The system will persist the user's choices for columns to be displayed per list type, and will display the same columns the next time the list is displayed. Preferences will be stored as cookies on the user's machine.

4.3.4.9 System (Actor)

The System actor represents any software component of the CHART system. It is used to model uses of the system which are either initiated by the system on an interval basis, or are an indirect by-product of another use cases that another actor has initiated.

4.3.4.10 Transfer Resources (Use Case)

The system allows the user to transfer controlled resources to another operations center that has logged in users. This is usually done when the last user at an operations center needs to log out but still has resources for which they are responsible. This feature provides a way for the user to hand off responsibility to another center, allowing the user to be permitted to log out. (The system prevents the last user from logging out of a center if that center has controlled resources.)

4.3.4.11 Transfer Traffic Event (Use Case)

The user can transfer a traffic event from their center to another center. In R3B3, this feature is enhanced to change the owning organization of the traffic event to the owning organization of the operations center to which the event is transferred.

4.3.4.12 View Devices Near Event (Use Case)

The system shall allow users with appropriate rights to view devices that are located close to a traffic event. This will appear as a separate section in the traffic event details page that is initially hidden, and can be expanded to show a heading for each device type, each of which can be expanded to show the devices of that type. The following device types are included: DMS, HAR, Detector, and Camera (including their local monitor usage). Details for each device (as available) will be shown as follows: DMS: Name/Location Description, Distance from the event, Route, Direction, Intersecting Feature, Current Message, Current Beacon State, Current Mode, and Current Status; HAR: Name/Location Description, Distance from the event, Route, Direction, Intersecting Feature, Current Mode, Current Status, and Current Transmitter Status; Detector: Name/Location Description, Distance from the event, Route, Direction, Intersecting Feature, Current Mode, Current Status, and Average Speed. The display of average speed will be dependent on the user's rights and the detector's owning organization, so for each detector the user may see the average speed, average speed range, or no data; Camera: Name/Location Description, Distance from the event, Route, Direction, Intersecting Feature, Current Mode, Current Status, Local monitors where displayed, and controlling operations center. This feature will only be available for traffic events that have a latitude and longitude specified, and for devices that have latitude and longitude defined. External devices that were imported into the CHART system will be included in the list of close devices if they fall within the specified radius of the event and will be marked such that the user can distinguish CHART devices from External devices.

4.3.4.13 View Lane Closed Percent (Use Case)

The user can view the percent of lanes closed for a traffic event while viewing a traffic event list. By default, this column will be hidden and the user must explicitly choose to display this column. Sorting and filtering will be permitted on this field.

4.3.4.14 View Network Connection Site (Use Case)

The user can view the network connection site of a traffic event while viewing a traffic

event list. By default, this column will be hidden and the user must explicitly choose to display this column. Sorting and filtering will be permitted on this field.

4.3.4.15 View Time Last Modified (Use Case)

The user can view the time a traffic event was last modified while viewing a traffic event list. By default, this column will be hidden and the user must explicitly choose to display this column. Sorting and filtering will be permitted on this field.

4.3.4.16 View Time Opened (Use Case)

The user can view the time a traffic event was opened while viewing a traffic event list. By default, this column will be hidden and the user must explicitly choose to display this column. Sorting and filtering will be permitted on this field.

4.3.4.17 View Traffic Event List (Use Case)

The ability to view a list of traffic events is enhanced in R3B3 to allow users to choose to show / hide columns. A couple of new columns are also being added that previously were sort columns (via a list box at the top of the page) but were not viewable. The traffic event list will restrict access to information based on a user's rights and the organization of the traffic event. The restricted information will include the ability to see external events, and the use of the word "fatality" in the traffic event name and incident type.

4.3.4.18 View Traffic Event Details (Use Case)

A user with appropriate rights shall be permitted to view the details for an event. This feature exists in R3B2 and is being enhanced in R3B3 to show new fields, such as the owning organization. The event details will be changed to restrict access based on user rights and the organization of the traffic event. The restricted information will include the use of the word "fatality" (in the name and incident type) and access to viewing the event's history log.

4.3.5 R3B3ManageDevices (Use Case Diagram)

This diagram shows uses of the system related to DMS and TSS Communication. For R3B3, TCP/IP device communications is supported for DMS's and HARS. In addition, R3B3 adds support for NTCIP compliant DMS messages.

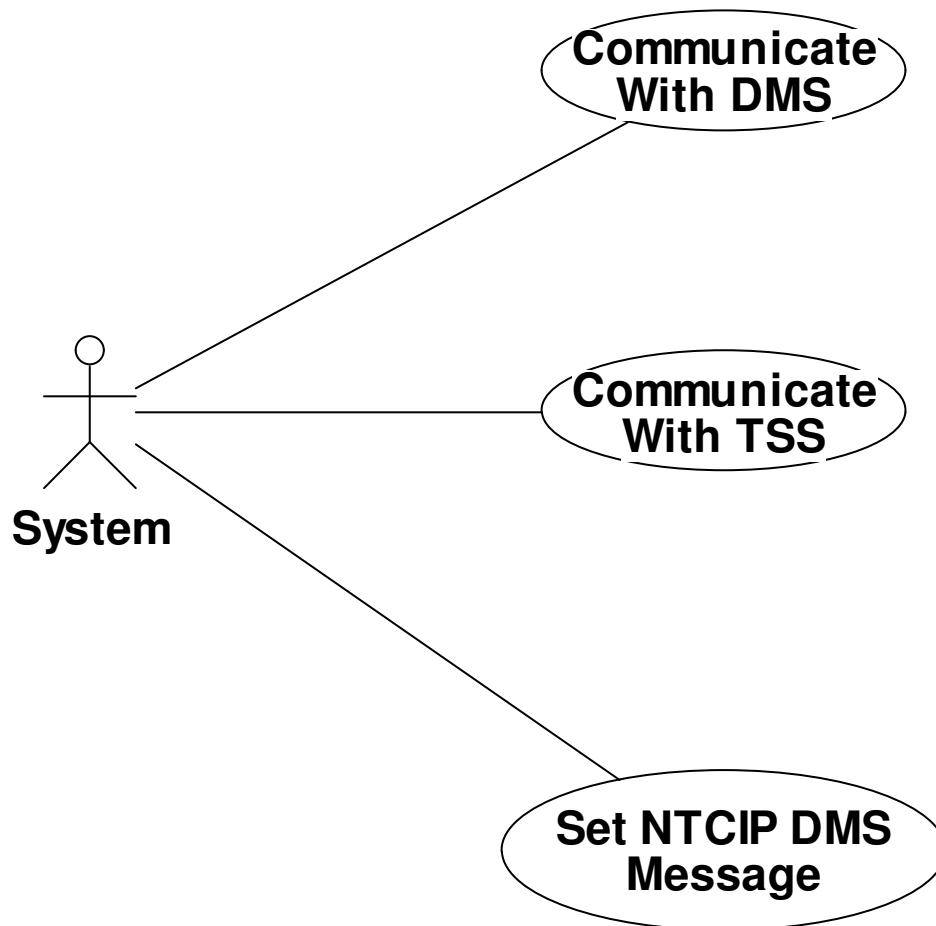


Figure 4-11 R3B3ManageDevices (Use Case Diagram)

4.3.5.1 Communicate With DMS (Use Case)

The system will support TCP/IP communication with a DMS that supports it, using the IP address and port number specified in the device configuration.

4.3.5.2 Communicate With TSS (Use Case)

The system will support TCP/IP communication with a TSS that supports it, using the IP address and port number specified in the device configuration.

4.3.5.3 Set NTCIP DMS Message (Use Case)

The system will automatically set the default font number and the default line spacing (as specified in the DMS configuration) for an NTCIP DMS before setting the message.

4.3.5.4 System (Actor)

The System actor represents any software component of the CHART system. It is used to model uses of the system which are either initiated by the system on an interval basis, or are an indirect by-product of another use cases that another actor has initiated.

4.4 External Interfaces (RITIS Import)

4.4.1 R3B3ImportRITISData (Use Case Diagram)

This diagram shows use cases related to RITIS. New for R3B3 is the import of external DMS/TSS's. Additionally, R3B3 provides enhanced Traffic Event import and connection monitoring.

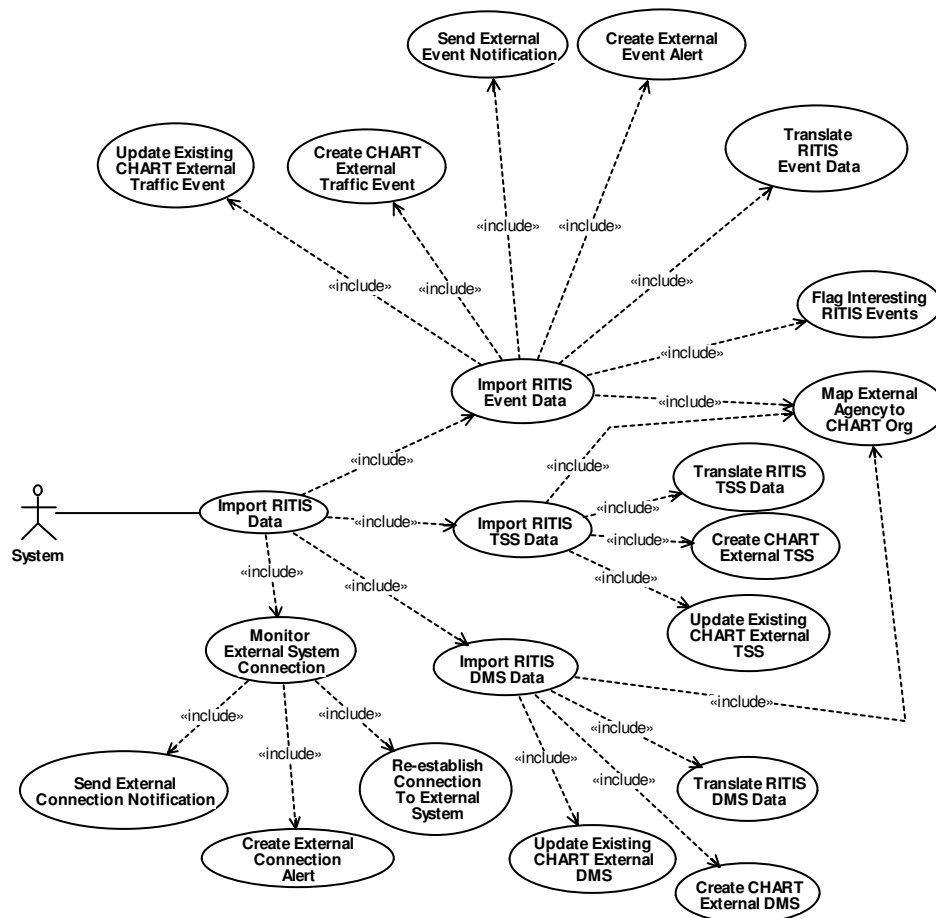


Figure 4-12 R3B3ImportRITISData (Use Case Diagram)

4.4.1.1 Create CHART External Traffic Event (Use Case)

The system shall create a new CHART External Event when a candidate event passes one of the configured event import rules. The data imported for an event shall include the following, as available in the external data source: event creation time, event closed time, scene cleared time, event state, external system name, external event id, external agency, location description, CHART event type, lane status, vehicles involved, and geolocation

data (lat/long).

4.4.1.2 Create CHART External DMS (Use Case)

When DMS data is imported from RITIS, if an administrator has already indicated that the DMS is to be included in CHART, and it has not already been added to CHART, it will be added to CHART as an External DMS.

4.4.1.3 Create CHART External TSS (Use Case)

When TSS data is imported from RITIS, if an administrator has already indicated that the TSS is to be included in CHART, and it has not already been added to CHART, it will be added to CHART as an External TSS.

4.4.1.4 Create External Connection Alert (Use Case)

The system will create an External Connection Alert if an external connection transitions to the "failed" state and remains there for an amount of time, as specified in the connection settings, if the alert is enabled in the connection settings. The system will create an External Connection Alert if an external connection transitions to the "warning" state and remains there for an amount of time, as specified in the connection settings, if the alert is enabled in the connection settings. (The time spent in the "failed" state contributes to the time counted for a warning).

4.4.1.5 Create External Event Alert (Use Case)

The system will create an External Event Alert for the operations center specified in the event import rule if an external traffic event matches an event import rule and the rule's settings indicate that an alert should be issued.

4.4.1.6 Flag Interesting RITIS Events (Use Case)

The system shall set the CHART event "interesting" flag when importing a RITIS event if an event import rule was configured to generate an alert and the rule passed.

4.4.1.7 Import RITIS Data (Use Case)

The system shall import data from the RITIS system.

4.4.1.8 Import RITIS DMS Data (Use Case)

The system shall import DMS data from RITIS. RITIS provides the data in the TMDD standard. Only DMSs that have been selected for inclusion in CHART by the administrator will become CHART external DMSs. Once a DMS has been included in CHART, its data will be kept updated in CHART as updates for it are received from RITIS.

4.4.1.9 Import RITIS Event Data (Use Case)

The system shall import event data from RITIS. The data is provided by RITIS in the SAE ATIS J2354 standard. This capability exists in R3B2, however it is being enhanced in R3B3. External event import rules, configured by an administrator, will be used to determine which events get imported into CHART. An event that meets any rule will be imported, events that do not match any rules will not get imported.

4.4.1.10 Import RITIS TSS Data (Use Case)

The system shall import TSS data from RITIS. RITIS provides the data in the TMDD standard. Only TSSs that have been selected for inclusion in CHART by the administrator will become CHART external TSSs. Once a TSS has been included in CHART, its data will be kept updated in CHART as updates for it are received from RITIS.

4.4.1.11 Map External Agency to CHART Org (Use Case)

The system shall translate the external agency ID for data imported from an external system into a CHART Organization. This mapping is done when the external object (Ex. Event, DMS, TSS) is created. (This is required to allow CHART to enforce organization based user rights.) The mapping of agency to organization will be configured by a system administrator. If an external object is received and an organization mapping has not been configured for its agency, a default organization will be used.

4.4.1.12 Monitor External System Connection (Use Case)

The system shall monitor the state of the External System Connection.

4.4.1.13 Re-establish Connection To External System (Use Case)

The system shall attempt to re-establish connection to the External System when the connection is down or unreasonably inactive.

4.4.1.14 Send External Connection Notification (Use Case)

The system will send a notification to a specified notification group if an external connection transitions to the "failed" state and remains there for an amount of time, as specified in the connection settings, if enabled in the connection settings. The system will send a notification to a specified notification group if an external connection transitions to the "warning" state and remains there for an amount of time, as specified in the connection settings, if enabled in the connection settings. (The time spent in the "failed" state contributes to the time counted for a warning).

4.4.1.15 Send External Event Notification (Use Case)

The system will send a notification to the group specified in the event import rule if an external traffic event matches an event import rule and the rule's settings indicate that a

notification should be sent.

4.4.1.16 System (Actor)

The System actor represents any software component of the CHART system. It is used to model uses of the system which are either initiated by the system on an interval basis, or are an indirect by-product of another use cases that another actor has initiated.

4.4.1.17 Translate RITIS DMS Data (Use Case)

When DMS data is imported from RITIS, the system will convert the data from the RITIS format (TMDD with RITIS extensions) into CHART format.

4.4.1.18 Translate RITIS Event Data (Use Case)

When an External Event is created or updated the system shall translate RITIS event data (SAE ATIS J2354) to CHART Traffic Event data. For R3B3 lane data, vehicles involved data and geolocation have been added to existing R3B2 translations.

4.4.1.19 Translate RITIS TSS Data (Use Case)

When TSS data is imported from the RITIS system, it will be translated from the format used by RITIS (TMDD with RITIS extensions) to CHART format.

4.4.1.20 Update Existing CHART External Traffic Event (Use Case)

The system shall update any existing CHART External Events (I.E. previously imported) with new information provided about that event from the external source. This includes updating CHART External Traffic Events to the closed state. The data updated for an event shall include the following, as available in the external data source: event creation time, event closed time, scene cleared time, event state, location description, event description, CHART event type, lane status, vehicles involved, and geolocation data (lat/long).

4.4.1.21 Update Existing CHART External DMS (Use Case)

When a DMS is imported from RITIS, if it has been previously added to CHART as an External DMS, the data for that external DMS will be updated.

4.4.1.22 Update Existing CHART External TSS (Use Case)

When a TSS is imported from RITIS, if it has been previously added to CHART as an External TSS, the data for that external TSS will be updated.

4.4.2 R3B3ConfigureExternalSystemSettings (Use Case Diagram)

This diagram shows use cases related to external system configuration settings.

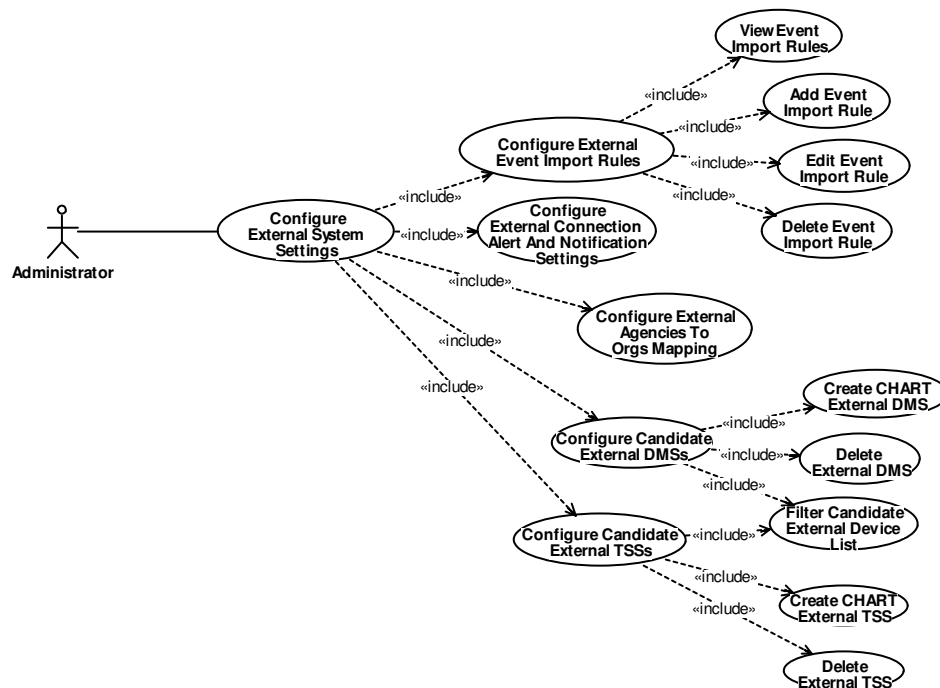


Figure 4-13 R3B3ConfigureExternalSystemSettings (Use Case Diagram)

4.4.2.1 Add Event Import Rule (Use Case)

A suitably privileged user can add an event import rule to the system. Each rule contains criteria that when met by an external event will cause that external event to be imported into the CHART system. Rules can also have actions that are taken if the criteria is met, including whether or not an External Event Alert should be sent and to which operations center, whether or not a notification should be sent and to which notification group, and whether or not the event's interesting flag should be set upon import. The following criteria are included:

Geographical Area: zero or more named geographical areas that have been defined in CHART. If an event's latitude/longitude falls within a specified geographical area, it matches the criteria. This criteria can be set to "empty" to indicate it matches events without a latitude/longitude, or "any" to indicate that any event (regardless of lat/long value) matches this criteria.

U.S. State: One or more States within the U.S. If an event contains a state and it is the same as the state criteria specified, it matches the criteria. This criteria can be set to "empty" to indicate it matches events that have no state specified, or "any" to indicate any event

matches regardless of its state value.

Route Type: A type of route, such as interstate, state road, county road, etc. as defined in CHART. An event with a route type specified that is the same as the route type specified in the rule criteria matches the criteria. A value of "empty" can be used to match events that don't have a route type specified, and the value of "any" can be used to cause any event to match regardless of its route type value.

Search Text: Zero or more text strings that must be found in one of the following fields in the event - event name, event description, route number, or county. If the search text criteria is found anywhere in one of these fields of the event, the event matches this criteria.

Lanes Closed: The number of lanes that must be indicated as closed in the event. A value of "empty" can be used to cause an event without lane closure data to match, and a value of "any" can be used to cause any event to match regardless of its lane closure data.

CHART Event Type: Zero or more CHART event types, such as incident, disabled vehicle, etc. An event whose event type maps to any of the selected CHART event types will match this criteria. All events will match this criteria if no event types are selected.

4.4.2.2 Administrator (Actor)

An administrator is a CHART user that has functional rights assigned to allow them to perform administrative tasks, such as system configuration and maintenance.

4.4.2.3 Configure Candidate External DMSs (Use Case)

The administrator can view a list of DMSs that exist in external systems (candidate external DMSs) and configure whether or not they are to be included in the CHART system. Because the list of devices could be large, a filtering capability is provided to allow the administrator to manage smaller groups of devices at one time. When showing a list of candidate DMSs (full list or filtered), the following fields will be included: DMS name, agency, location description, indication of whether the DMS is currently marked for inclusion in CHART, and an indication of whether the DMS is currently marked for exclusion from CHART.

4.4.2.4 Configure Candidate External TSSs (Use Case)

The administrator can view a list of TSSs that exist in external systems (candidate external TSSs) and configure whether or not they are to be included in the CHART system. Because the list of devices could be large, a filtering capability is provided to allow the administrator to manage smaller groups of devices at one time. When showing a list of candidate TSSs (full list or filtered), the following fields will be included: TSS name, agency, location description, indication of whether the TSS is currently marked for inclusion in CHART, and an indication of whether the TSS is currently marked for exclusion from CHART.

4.4.2.5 Configure External Agencies To Orgs Mapping (Use Case)

The administrator can manage the list of external agencies that may exist in data exported from external systems, and set the CHART organization to be assigned to external objects from that agency during import. Each agency will be specified for use by a specific external system and should be unique within that external system.

4.4.2.6 Configure External Connection Alert And Notification Settings (Use Case)

The system will allow a suitably privileged user to configure the settings for external system connection alerts and notifications. For each external system the user will be able to specify a time threshold for sending alerts/notifications if the connection is in a warning or failure state for at least the specified period of time. The user will be able to specify whether warning alerts/notifications should be sent in addition to failures. The user will be able to specify whether alerts are enabled or disabled, and (if enabled) which operations center to send the alerts to. The user will be able to specify whether notifications are enabled or disabled, and (if enabled) which group to send notifications to.

4.4.2.7 Configure External Event Import Rules (Use Case)

A user with sufficient rights will be able to add, edit, and remove rules for importing external events. These rules will contain criteria that an external event must meet to be imported into the CHART system. Each rule will also specify whether the event is to be marked "interesting", should generate an External Event Alert, or should cause a notification message to be sent when the event is imported.

4.4.2.8 Configure External System Settings (Use Case)

The system shall allow a user with appropriate rights to configure settings related to external systems. Several settings exist in R3B2, and R3B3 will add additional settings. External Event Rules will govern which external events are imported into CHART, whether they are marked as "interesting", whether an External Event Alert is generated for them, and whether a notification is sent. The user can also choose which external DMSs and detectors to include / exclude from CHART. The user will be able to specify connection failure timeouts for sending External Connection Alerts and/or notifications. The user will be able to specify the external system/agency mapping to CHART organizations, for assigning an organization when importing traffic events and devices.

4.4.2.9 Create CHART External DMS (Use Case)

When DMS data is imported from RITIS, if an administrator has already indicated that the DMS is to be included in CHART, and it has not already been added to CHART, it will be added to CHART as an External DMS.

4.4.2.10 Create CHART External TSS (Use Case)

When TSS data is imported from RITIS, if an administrator has already indicated that the

TSS is to be included in CHART, and it has not already been added to CHART, it will be added to CHART as an External TSS.

4.4.2.11 Delete External DMS (Use Case)

The system shall allow a suitably privileged user to delete an external DMS from the CHART system. When doing so, the DMS shall remain as a candidate DMS that is marked as being excluded from the CHART system. The administrator can choose to include the DMS at a later time.

4.4.2.12 Delete Event Import Rule (Use Case)

A suitably privileged user can delete an event import rule from the system. The system shall require the user to confirm this action before it is completed.

4.4.2.13 Delete External TSS (Use Case)

The system shall allow a suitably privileged user to delete an external TSS from the CHART system. When doing so, the TSS shall remain as a candidate TSS that is marked as being excluded from the CHART system. The administrator can choose to include the TSS at a later time.

4.4.2.14 Edit Event Import Rule (Use Case)

A suitably privileged user can edit an existing event import rule. This includes changing any of the filter criteria or actions.

4.4.2.15 Filter Candidate External Device List (Use Case)

The user may use a filter when displaying a list of external devices that are candidates for inclusion in the CHART system. The following filter criteria are included:

Agency: zero or more agencies used to show only devices that contain that agency

Geographical Area: zero or more named geographical areas defined in CHART used to show only devices whose lat/long falls within one of the specified areas

Text Search: zero or more text strings - only devices that contain one of the specified text strings within the device's name, description, location description, county, or route name will be shown

Included indicator: When set shows devices that have been selected for inclusion in CHART

Excluded indicator: When set shows devices that have been selected for exclusion from CHART

New indicator: When set shows devices that have not been selected for inclusion in OR

exclusion from CHART

4.4.2.16 View Event Import Rules (Use Case)

The system shall allow a suitably privileged user to view the event import rules that have been defined.

4.5 Public/Private Data Sharing

4.5.1 R3B3ManageUsers (Use Case Diagram)

This diagram shows uses of the system related to user management. The system has Roles defined by an Administrator that contain user rights. Users are assigned roles, and when a user logs into the system they are granted all rights contained in their assigned roles. For R3B3, several new user rights are being added that fall into the category of "public/private data sharing". They control access to data in the system that may not be desirable to distribute to some users of the CHART system. This may be due to the sensitive nature of the data, or due to contractual obligations that limit the right to distribute the data to third parties. These rights (as do all user rights) also determine the data that is accessible to a client of the CHART system's external interface (export web service), as roles are assigned to external systems similar to the way they are assigned to users.

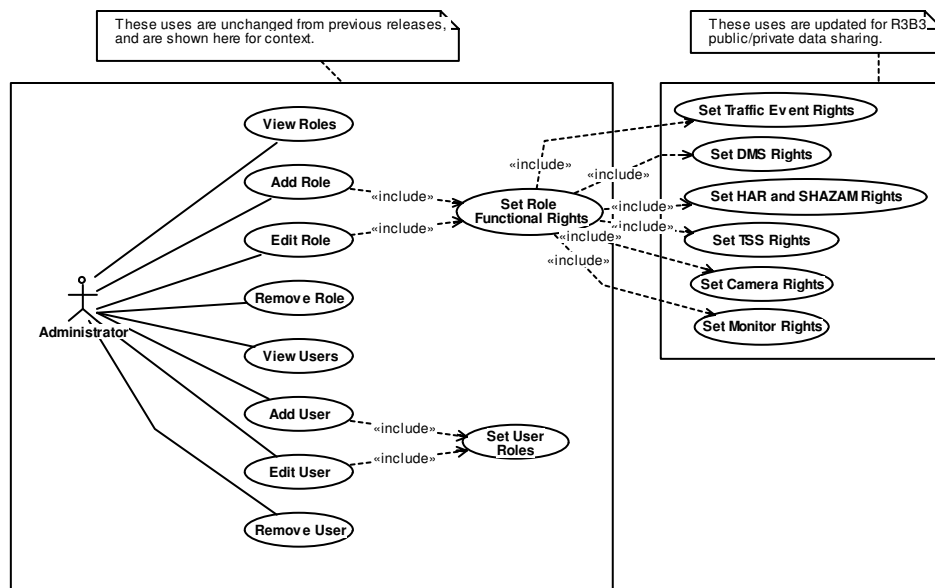


Figure 4-14 R3B3ManageUsers (Use Case Diagram)

4.5.1.1 Add Role (Use Case)

An administrator can add a new role to the system.

4.5.1.2 Add User (Use Case)

An administrator can add a new user to the system.

4.5.1.3 Administrator (Actor)

An administrator is a CHART user that has functional rights assigned to allow them to perform administrative tasks, such as system configuration and maintenance.

4.5.1.4 Edit Role (Use Case)

An administrator can edit an existing role.

4.5.1.5 Edit User (Use Case)

An administrator can edit an existing user.

4.5.1.6 Remove Role (Use Case)

An administrator can remove a role from the system.

4.5.1.7 Remove User (Use Case)

An administrator can remove a user from the system.

4.5.1.8 Set Camera Rights (Use Case)

An administrator can include rights associated with Cameras in a Role. For R3B3 the following rights are added: View Camera Sensitive Config: controls the ability to view sensitive camera configuration data. This right is assignable per organization. Sensitive configuration data for cameras depends on the type of sending and control device used by the camera as follows: Sending Device Type is Encoder: Sensitive data includes IP Video Fabric, Encoder Type, Port, Multicast Address, and Multicast Port. Sending Device Type is Switch: Sensitive data includes the switch and input port. Control Device Type is IP via Codec: Sensitive data includes the IP, Port, Baud, Bits, Parity, Stop Bits, and Flow Control. Control Device Type is Comm Port: Sensitive data includes Port, Baud, Parity, Stop Bits, and Flow Control. Control Device Type is Command Processor: Sensitive data includes the command processor name.

4.5.1.9 Set DMS Rights (Use Case)

An administrator can set the DMS related rights included in a Role. For R3B3, the following rights are being added: View External DMSs: controls the ability to view external DMSs in the DMS list; View DMS Sensitive Config: Controls the ability to view sensitive configuration data. This right can be granted per organization. The sensitive configuration data for a DMS includes the following communication settings: Drop Address, Port Manager Connection Timeout, Port Type, Baud, Data Bits, Parity, Stop Bits, Flow Control, Default Telephone Number (ISDN & POTS), and the per port manager telephone numbers.

4.5.1.10 Set HAR and SHAZAM Rights (Use Case)

An administrator can include rights associated with HARs and SHAZAMs in a Role. For R3B3 new rights are being added as follows: View HAR Sensitive Config: controls the ability to view sensitive HAR and SHAZAM configuration data. This right will be assignable per organization. Sensitive configuration data for a HAR or SHAZAM includes Default Phone Number, Access Code, Port Manager Connection Timeout, Port Type, and per port manager phone numbers.

4.5.1.11 Set Monitor Rights (Use Case)

An administrator can include rights associated with Monitors in a Role. For R3B3 the following rights are added: View Monitor Sensitive Config: controls the ability to view sensitive monitor configuration data. This right is assignable per organization. The configuration data considered sensitive depends on the type of receiving device used by the monitor as follows: Receiving Device of Type Decoder: Sensitive data includes IP Video Fabric, IP, Type, TCP Port, Video Port. Receiving Device of Type Switch: Sensitive data includes the Switch and output port.

4.5.1.12 Set Role Functional Rights (Use Case)

An administrator can set the functional rights included in a Role when adding a role or editing an existing role.

4.5.1.13 Set Traffic Event Rights (Use Case)

Traffic event related rights may be included in a Role. For R3B3, new rights are being added as follows: View External Traffic Events: controls ability for user to view external event list and external event tab on the home page; View Traffic Event Sensitive Incident Details: controls ability to see sensitive incident details, such as if the incident involves a fatality (in the name and the incident type); View Traffic Event Log: controls ability to view the traffic event log. This right can be controlled per organization such that a user can view some organization's traffic event's logs but not others.

4.5.1.14 Set TSS Rights (Use Case)

An administrator can include TSS (Detector) related rights in a Role. For R3B3, the following rights are added: View External TSS: controls the ability to view external TSSs in the TSS list; View TSS Sensitive Config: controls the ability to view sensitive configuration data. This right can be assigned per organization. The sensitive configuration data for a TSS includes the following communication settings: Drop Address, Port Manager Connection Timeout, Port Type, Baud, Data Bits, Parity, Stop Bits, Flow Control, Default Telephone Number (ISDN & POTS), and the per port manager telephone numbers; View VSO Detailed Data: controls the ability to view a detector's detailed volume, speed, and occupancy data. This right is assignable per organization. View VSO Summary Data: controls the ability to view a detector's summary VSO data (speed range). This right is assignable per organization and has no effect if the role also has the View VSO Detailed

Data right.

4.5.1.15 Set User Roles (Use Case)

An administrator can set the roles granted to a user while adding a user or editing an existing user.

4.5.1.16 View Roles (Use Case)

The administrator can view the roles defined in the system.

4.5.1.17 View Users (Use Case)

An administrator can view the users that exist in the system.

4.5.2 R3B3ProvideDataToExternalSystems (Use Case Diagram)

This diagram shows uses of the system related to providing data to external systems. For R3B3, CHART will provide Traffic Event, HAR, DMS, and SHAZAM to external systems. In addition R3B3 will provide external client management.

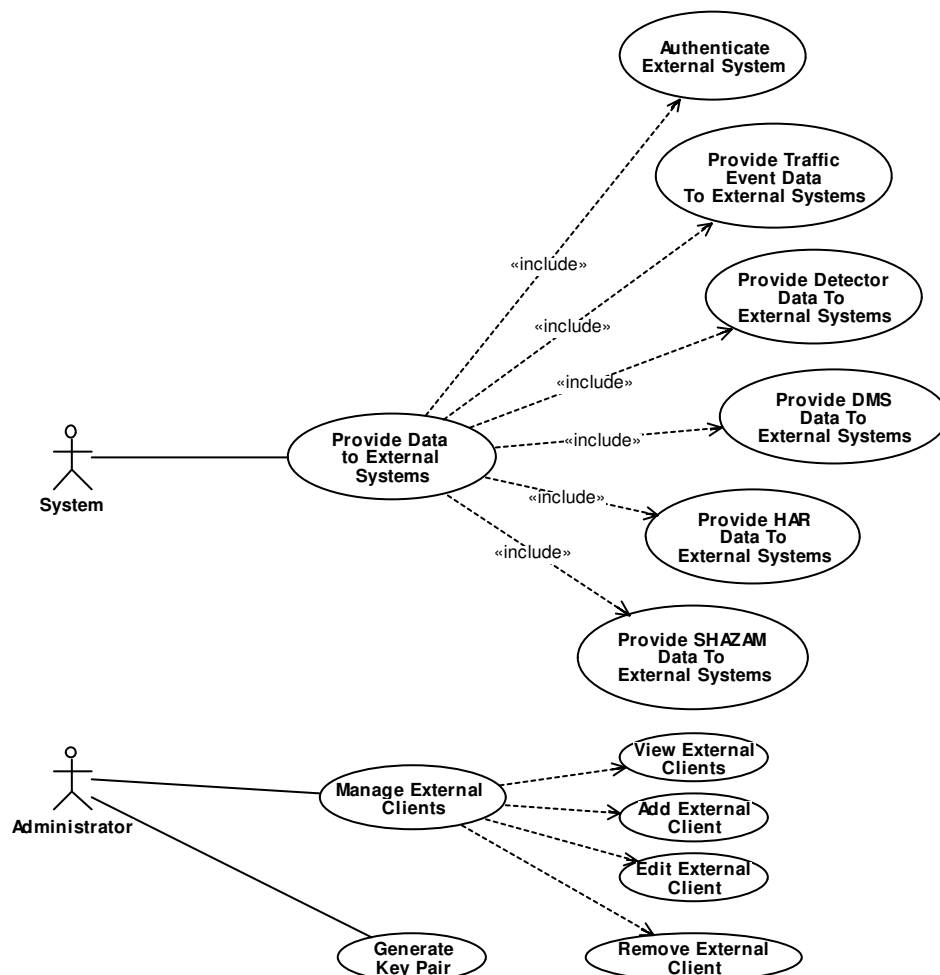


Figure 4-15 R3B3ProvideDataToExternalSystems (Use Case Diagram)

4.5.2.1 Add External Client (Use Case)

The system shall allow an administrator to add an external client to the CHART system. When doing so, the administrator will specify data pertaining to the client including a client ID to be used by the external system and a public key used to validate data signed by the external system. The administrator will specify whether the external system is a supplier and/or consumer of CHART data, and if it is a consumer, one or more CHART Roles whose user rights will determine the data accessible to the external system. The administrator will also be able to specify a name and description of the external system,

contact information for a person responsible for managing the external system.

4.5.2.2 Administrator (Actor)

An administrator is a CHART user that has functional rights assigned to allow them to perform administrative tasks, such as system configuration and maintenance.

4.5.2.3 Authenticate External System (Use Case)

The system shall authenticate external system connections to validate that they are authorized to connect to CHART and to enforce rights regarding the data the external system is permitted to access. Each external system owner will be provided a private key from a public/private key pair generated within the CHART system. Each request from an external system will include the system's CHART client ID (as configured within CHART) and a digital signature of the request data, created using the private key provided by the CHART administrator. The CHART system will validate each request signature using the client's public key.

4.5.2.4 Edit External Client (Use Case)

The system shall allow an administrator to edit the data associated with an external client, as described in the Add External Client use case.

4.5.2.5 Generate Key Pair (Use Case)

The system shall allow an administrator to generate a public/private key pair for use in controlling access to the CHART external interface. The private key is to be given (offline) to the owner of the external system wishing to gain access to CHART data. The public key is to be used by the administrator when adding the external client to the CHART system that corresponds to the external system that wishes to retrieve data from CHART.

4.5.2.6 Manage External Clients (Use Case)

The system shall allow an administrator to manage the external clients that are permitted to retrieve data from CHART via its external system interface.

4.5.2.7 Provide Data to External Systems (Use Case)

The system shall provide access to external systems via a web service to allow them to receive data that the CHART system makes available to third parties. One or more Roles assigned to each external client will be used to determine the data the client will be permitted to access. All requests made by external systems shall be validated against published XSD. CHART will return a response XML document for each request. The XML returned will contain an error code and error text for invalid requests, and will return the requested data for valid, authorized requests. The response XML shall be formatted as specified in published XSD.

4.5.2.8 Provide Detector Data To External Systems (Use Case)

The system shall provide detector data to external systems. The system shall enforce granular, organization based user rights to allow the level of detail provided for a detector to be controlled. Two user rights will be used to determine if a detector's detailed volume, speed, and occupancy (VSO) data is exported, only a speed range, or no VSO data. When VSO data is provided for a detector, it will include the data for zone groups and for each zone within the group. The detector data will be provided using the TMDD standard, with CHART extensions as needed. External systems can obtain an inventory of detectors that includes each detector's current status, and can obtain updates to the detector data (including the status) by requesting a new inventory list.

4.5.2.9 Provide DMS Data To External Systems (Use Case)

The system shall allow external systems to receive data pertaining to DMSs using the TMDD standard, with CHART extensions to the standard as needed. External systems can obtain an inventory of DMS devices, including their current status. Updates can be obtained by obtaining a new inventory.

4.5.2.10 Provide HAR Data To External Systems (Use Case)

The system shall provide HAR data to external systems using the TMDD standard format, with CHART extensions as needed. External systems can obtain an inventory of HARs, including the HAR current status. External systems can receive updates by requesting a new inventory.

4.5.2.11 Provide SHAZAM Data To External Systems (Use Case)

The system shall provide SHAZAM (beacon) data to external systems using the TMDD standard format, with CHART extensions as needed. External systems can obtain an inventory of SHAZAMs, including the SHAZAM current status. External systems can receive updates by requesting a new inventory.

4.5.2.12 Provide Traffic Event Data To External Systems (Use Case)

The system shall allow external systems to receive traffic event data from CHART. The CHART system shall enforce granular, organization based use rights to determine the level of detail that may be seen by each event. User rights will control whether or not the incident type of "fatality" is exported within the event name and the incident type field. A cleansed version of the incident type and event name that substitutes personal injury for fatality will be exported to external clients that don't have the proper user right. A separate user right determines whether or not event history for an event is exported. The traffic event data is exported using the SAE ATIS J2354 standard format with CHART extensions as needed. External systems can obtain a list of traffic events (an inventory) and can receive updates by polling to receive a new inventory periodically.

4.5.2.13 Remove External Client (Use Case)

The system shall allow an administrator to remove an external client from the system, effectively preventing them from accessing CHART's external interface to retrieve data. The system will prompt the user for confirmation before removing the client.

4.5.2.14 System (Actor)

The System actor represents any software component of the CHART system. It is used to model uses of the system which are either initiated by the system on an interval basis, or are an indirect by-product of another use cases that another actor has initiated.

4.5.2.15 View External Clients (Use Case)

The system shall allow an administrator to view the external clients allowed to retrieve CHART data via its external interface.

4.6 Alerts/Notifications

4.6.1 R3B3ManageAlertsAndNotifications (Use Case Diagram)

This diagram shows use cases related to alerts and notifications. For R3B3, three new alert types are added. Additionally, there are 3 conditions where notifications can be sent automatically by the system (if configured to do so). All existing (pre-R3B3) use cases related to managing alerts apply to the new alert types.

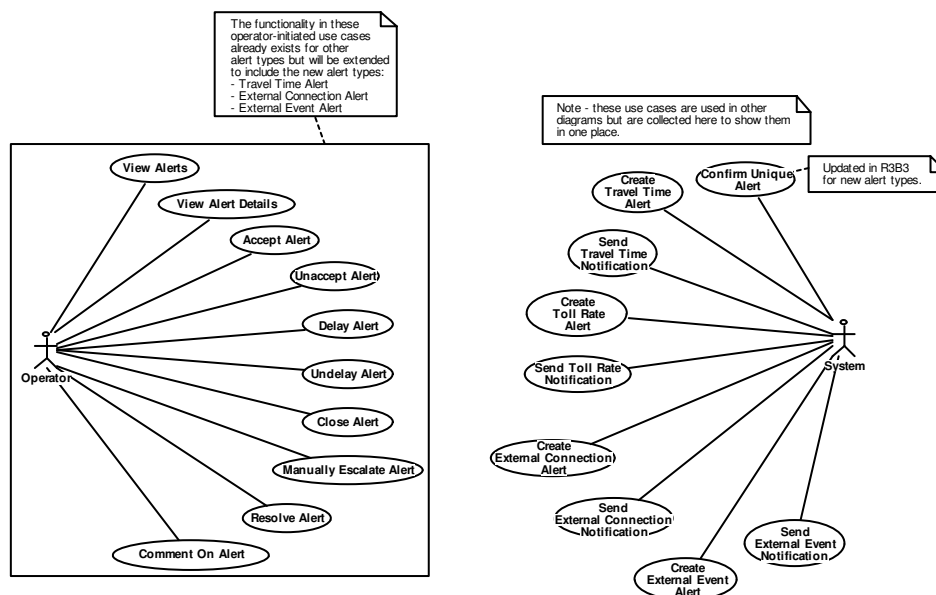


Figure 4-16 R3B3ManageAlertsAndNotifications (Use Case Diagram)

4.6.1.1 Accept Alert (Use Case)

A user with sufficient privileges may Accept an alert. Accepting an alert implies the user's AMG will handle the alert to closure. Accepting an alert stops any Escalation or Delay Timer, if running. To ensure alerts do not get accepted and forgotten, Accepting an alert starts the Accept timer for when the system should automatically revert the alert to the New state (See Configure Alert Timeouts). A typical duration of the Accept timer is expected to be less than a typical duration of the Delay timer.

4.6.1.2 Close Alert (Use Case)

A user with sufficient privileges may close an alert in the New, Accepted, or Delayed states. Closing an alert stops any Escalation, Delay, or Accept Timer and starts an Archive Timer. The alert remains visible to privileged viewers for the duration of the Archive Timer. After the Archive Timer expires the alert is removed from being seen by operators

and only exists in the database archives.

4.6.1.3 Comment On Alert (Use Case)

A user with proper functional rights can add a comment to an alert. Previous comments cannot be changed or removed, nor can the text used to create the alert be changed, but any appropriate comment can be attached to the alert. The comment will be time stamped, attributed to the user, stored in the Alert History in chronological order with other history entries.

4.6.1.4 Confirm Unique Alert (Use Case)

The system ensures duplicate non-closed alerts are not seen by the users. A duplicate alert is defined as two alerts with the same alert type and the same discriminator based on the alert type. The alert discriminators are as follows: EventStillOpenAlert: same event; DuplicateEventAlert: same event; UnhandledResourceAlert: same resource; ManualAlert: same alert description; DeviceFailureAlert: same device; ExecuteScheduledActionsAlert: same list of actions; External Connection Alert: same external connection; External Event Alert: same external event; Travel Time Alert: same travel route. Toll Rate Alert: Same travel route, same alert description.

4.6.1.5 Create External Connection Alert (Use Case)

The system will create an External Connection Alert if an external connection transitions to the "failed" state and remains there for an amount of time, as specified in the connection settings, if the alert is enabled in the connection settings. The system will create an External Connection Alert if an external connection transitions to the "warning" state and remains there for an amount of time, as specified in the connection settings, if the alert is enabled in the connection settings. (The time spent in the "failed" state contributes to the time counted for a warning).

4.6.1.6 Create External Event Alert (Use Case)

The system will create an External Event Alert for the operations center specified in the event import rule if an external traffic event matches an event import rule and the rule's settings indicate that an alert should be issued.

4.6.1.7 Create Toll Rate Alert (Use Case)

If toll rate alerts are enabled for a travel route, the system will issue an alert to the operations center specified in the travel route settings if the current toll rate for the travel route expires while there is a current (non-expired) toll rates document available.

4.6.1.8 Create Travel Time Alert (Use Case)

If travel time alerts are enabled for a travel route, the system will issue an alert to the operations center specified in the travel route settings when the travel time crosses the

threshold specified in the travel route settings.

4.6.1.9 Delay Alert (Use Case)

A user with sufficient privileges may Delay an alert. The implication is that the AMG is not going to handle the alert any time soon but still wants to take responsibility for handling the alert to closure. Delaying an alert stops any Escalation or Accept Timer, if running. To ensure alerts do not get delayed and forgotten, Delaying an alert starts the Delay Timer for when the system should automatically revert the alert to the New state (See Configure Alert Timeouts). A typical duration of the Delay timer is expected to be more than the typical duration of the Accept timer.

4.6.1.10 Manually Escalate Alert (Use Case)

A user with proper functional rights can force escalation of an alert. This performs an escalation cycle, which, if possible, adds additional operations centers (or in a future release, Areas of Responsibility) to the visibility of the alert.

4.6.1.11 Operator (Actor)

An operator is a user of the system who has been assigned a valid username/password combination and granted roles for system access.

4.6.1.12 Resolve Alert (Use Case)

A user with sufficient privileges may resolve alerts. Resolving an alert brings the user to a page where this type of alert can be addressed. The following resolve pages are envisioned for current alerts: DeviceFailure: Device Details page to allow the device to be taken offline or put into maintenance mode, if appropriate; UnhandledResource: Transfer Shareable Resource page; EventStillOpenAlert: Event Details page; DuplicateEventAlert: Merge Events page; ManualAlert: Alert details page; ExecuteScheduledActionsAlert: For schedules containing one action, an action specific page will be displayed to the user. Resolving an OpenEventAction will display the Pending Event's page. For multiple scheduled actions the execute scheduled action page for the schedule will be displayed. For a schedule with no actions the ExecuteScheduledActionsAlert's details page will be displayed; External Connection Alert: External connection status list page; External Event Alert: Event details page for the external event; Travel Time Alert: Travel route details page. Toll Rate Alert: Travel route details page.

4.6.1.13 Send External Connection Notification (Use Case)

The system will send a notification to a specified notification group if an external connection transitions to the "failed" state and remains there for an amount of time, as specified in the connection settings, if enabled in the connection settings. The system will send a notification to a specified notification group if an external connection transitions to the "warning" state and remains there for an amount of time, as specified in the connection settings, if enabled in the connection settings. (The time spent in the "failed" state

contributes to the time counted for a warning).

4.6.1.14 Send External Event Notification (Use Case)

The system will send a notification to the group specified in the event import rule if an external traffic event matches an event import rule and the rule's settings indicate that a notification should be sent.

4.6.1.15 Send Toll Rate Notification (Use Case)

If toll rate notifications are enabled for a travel route, the system will send a notification to the group specified in the travel route settings if the current toll rate for the travel route expires while there is a current (non-expired) toll rates document available.

4.6.1.16 Send Travel Time Notification (Use Case)

If travel time notifications are enabled for a travel route, the system will send a notification to the group specified in the travel route settings when the travel time crosses the threshold specified in the travel route settings.

4.6.1.17 System (Actor)

The System actor represents any software component of the CHART system. It is used to model uses of the system which are either initiated by the system on an interval basis, or are an indirect by-product of another use cases that another actor has initiated.

4.6.1.18 Unaccept Alert (Use Case)

A user with sufficient privileges may unaccept an alert in the Accepted state. Unaccepting an alert stops the Accept Timer, puts the alert in the New state, and begins the Escalation Timer. Unaccepting an alert implies that the user's AMG has changed their mind and no longer wishes to handle the alert.

4.6.1.19 Undelay Alert (Use Case)

A user with sufficient privileges may undelay an alert in the Delayed state. Undelaying an alert stops the Delay Timer, puts the alert in the New state, and begins the Escalation Timer. Undelaying an alert implies that the user's AMG has changed their mind and no longer wishes to handle the alert.

4.6.1.20 View Alert Details (Use Case)

A user with sufficient privileges may view alert details including the alert type, alert description, create time, next escalation time (if New), Unaccept time (if Accepted), Undelay time (if Delayed), closed time (if Closed), the current set of AMGs, the predicted set of AMGs at next escalation, and a history of all modifications to the alert each with a comment.

4.6.1.21 View Alerts (Use Case)

A user with sufficient privileges may view alerts. Viewing an alert includes the ability to see the alert type, the alert description, and the alert creation time. Alerts are organized by their state including an indication of the number of alerts in each state. A visual and auditory cue is given when the user is a member of an AMG listed in at least one New alert and the user has the rights to control the alert. The ability to view alerts does not imply the ability to control alerts. Closed alerts may be viewed only if they have not yet been archived.

4.7 Configure system

4.7.1 R3B3ConfigureSystem (Use Case Diagram)

This use case diagram shows use cases related to system configuration.

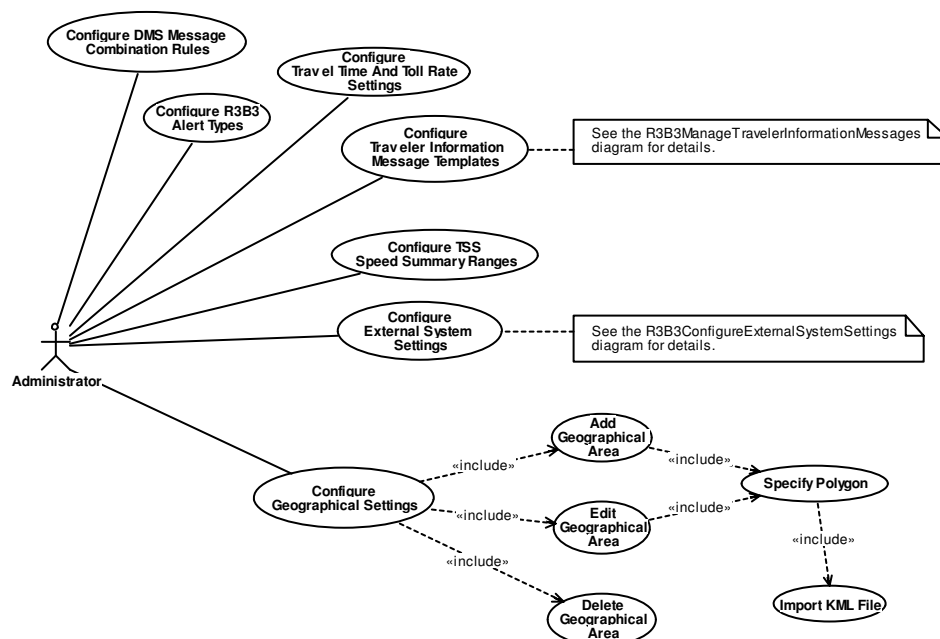


Figure 4-17 R3B3ConfigureSystem (Use Case Diagram)

4.7.1.1 Add Geographical Area (Use Case)

A user with sufficient rights will be able to add a geographical area to the system. The user will be able to specify a name and a polygon to represent the area.

4.7.1.2 Administrator (Actor)

An administrator is a CHART user that has functional rights assigned to allow them to perform administrative tasks, such as system configuration and maintenance.

4.7.1.3 Configure Travel Time And Toll Rate Settings (Use Case)

The system shall allow users with appropriate rights to set configuration values related to travel times. The user will be able to specify a system-wide display schedule for travel time DMS messages. The user will be able to specify the travel time range settings, which consist of boundaries for classifying an actual travel time into buckets, and for each bucket, values to add/subtract from the actual travel time to obtain the range to display. The user will be able to specify a threshold percentage for determining whether the travel time trend is classified as "up", "down", or "flat".

4.7.1.4 Configure DMS Message Combination Rules (Use Case)

The system shall allow a user with appropriate rights to set the message combination rules for DMSs. This feature exists in R3B2 and will be enhanced in R3B3 to support travel time and toll rate messages.

4.7.1.5 Configure External System Settings (Use Case)

The system shall allow a user with appropriate rights to configure settings related to external systems. Several settings exist in R3B2, and R3B3 will add additional settings. External Event Rules will govern which external events are imported into CHART, whether they are marked as "interesting", whether an External Event Alert is generated for them, and whether a notification is sent. The user can also choose which external DMSs and detectors to include / exclude from CHART. The user will be able to specify connection failure timeouts for sending External Connection Alerts and/or notifications. The user will be able to specify the external system/agency mapping to CHART organizations, for assigning an organization when importing traffic events and devices.

4.7.1.6 Configure Geographical Settings (Use Case)

A user with sufficient rights will be able to specify the geographical settings. The user will be able to add, edit, and remove named geographical areas which can be used for importing external events and devices in R3B3 (and will be useful for defining areas of responsibility in a future release). The user will also be able to specify the bounds (north, south, east, and west boundaries) for acceptable geographic coordinates when coordinates are entered by the user.

4.7.1.7 Configure R3B3 Alert Types (Use Case)

The system will allow a suitably privileged user to configure these new alert types introduced in R3B3: External Connection Alert, External Event Alert, and Travel Time Alert. For each of these types, the user will be able to specify the default and maximum accept timeouts, the default and maximum delay timeouts, the escalation timeout, an enable/disable flag for the alert type, and a flag for enabling/disabling automatic escalation. The user will also be able to configure the alert audio cues for the new alert types.

4.7.1.8 Configure Traveler Information Message Templates (Use Case)

A user with sufficient rights will be able to configure the available DMS message templates that may be used for travel time / toll rate messages. The user will be able to add, edit, remove, and view these templates.

4.7.1.9 Configure TSS Speed Summary Ranges (Use Case)

A user with sufficient rights will be able to configure the speed ranges to display when a user has rights to view speed summary information for a detector, but does not have rights to view the details.

4.7.1.10 Delete Geographical Area (Use Case)

The system will allow a user with sufficient rights to remove a geographical area from the system. The system will ask for confirmation before deleting the area, and will not allow an area to be deleted if it is referenced in the system.

4.7.1.11 Edit Geographical Area (Use Case)

A user with sufficient rights will be able to edit an existing geographical area defined in the system. The user will be able to specify a name and a polygon to represent the area.

4.7.1.12 Import KML File (Use Case)

A user specifying the coordinates for a geographical area polygon will be able to import the polygon definition using a KML (Keyhole Markup Language) file. This XML-based file format is used by Google Earth and other tools.

4.7.1.13 Specify Polygon (Use Case)

A user adding or editing a geographical area will be able to specify the coordinates of a polygon using geographical (latitude/longitude) coordinates. The user will be able to enter the coordinates manually, or import the polygon definition from a Keyhole Markup Language (KML) file. If entering the coordinates manually, the user will be able to add/insert a point, edit an existing point, or remove a point. All coordinates that are manually entered will be validated against the system-wide bounds from the system profile to make sure the coordinates are not unreasonable.

5 Detailed Design

5.1 Human-Machine Interface

R3B3 builds on the existing web based user interface that has been used for the CHART system for past releases. A usable prototype of the proposed interface changes has been created in order to allow the user to preview the changes and to better facilitate a common understanding of the requirements. These changes fall into the areas of Travel Routes, DMS Message Templates, Travel Time and Toll Rate Messages, DMS Travel Time / Toll Rate Settings, System Profile Travel Time Settings, Geographical Settings, External Events, External Devices, External System Settings, External Connection Status, Alerts, Device Locations, Miscellaneous Device Enhancements, and Public/Private Data Sharing. See the sections below for details.

5.1.1 Travel Routes

Travel Routes are CHART objects that provide a building block for travel time and toll rate messages. Travel Routes represent segments of roadway and usually start at a DMS and end at a destination well known to travelers. A travel route may be associated with one or more roadway links which provide travel time data for the travel route. A travel route may also be associated with a toll rate source, which provides toll rate data for the travel route. Features exist in R3B3 to allow travel routes to be added, edited, and removed. Users can also view travel routes that have been added to the system along with their current travel time and/or toll rate.

5.1.1.1 Viewing Travel Routes

A new menu item will appear within the existing home page navigation area to allow users to view existing travel routes.

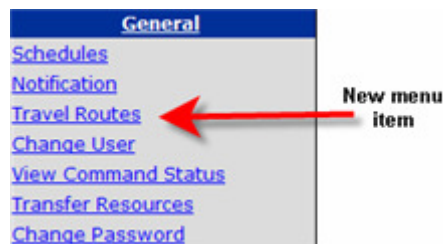


Figure 5-1 View Travel Routes menu item

Upon clicking the Travel Routes menu item, a page will be shown in the “working window” listing all travel routes that exist in the system (see below).

Travel Routes (7) [Add Travel Route](#) [Set Columns](#)

Name	Length	Trav Time	Trend	Speed	Toll Rate	Used By	Route	Dir	County	
		Any	Any	Any	Any		Any	Any	Any	
152 to White Marsh	5 mi	7 mins	Up	55	N/A	DMS XYZ	I-95	SB	Harford County	details edit remove
270 Split to Amer. Legion Br.	10 mi	15 mins	Down	55	N/A	DMS XYZ	I-95 I-495	SB OL	Montgomery County	details edit remove
ICC Exit 1 to Exit 3	7 mi			55	\$0.75 at 14:00	DMS XYZ	MD 200	EB	Montgomery County	details edit remove
ICC Exit 6 to Exit 8	4 mi	6 mins	Flat	55	\$0.50 at 14:05	DMS XYZ	MD 200	EB	Montgomery County Prince George's County	details edit remove
Route - no current TT data	4 mi				N/A	DMS XYZ	I-95	NB	Baltimore County	details edit remove
Route - no current Toll data	4 mi	N/A	N/A	N/A		DMS XYZ	MD 200	WB	Montgomery County	details edit remove
Route - tt and toll disabled	3 mi	N/A	N/A	N/A	N/A		MD 200	WB	Montgomery County	details edit remove

Top | Back | Forward | Refresh | Center Rpt | Communications Log | Instant Messaging | Home Page | Paging | Map | Traffic Events | Help | Save Window Position

CHART R3B3 Prototype 11/12/2008 © 2002-2008 MDSHA. All rights reserved.

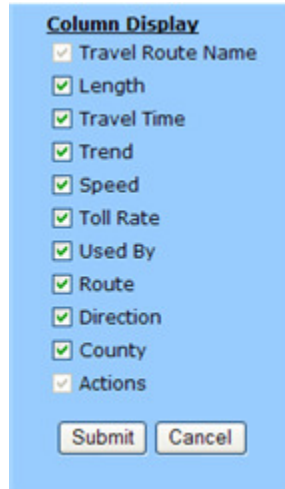
Local intranet 100%

Figure 5-2 View Travel Routes page

This page shows the following information for each travel route:

- Travel Route Name
- The length of the route (in miles)
- The current travel time for the route (if any)
- The current travel time trend (if any)
- The current average speed on the route (if any)
- The current toll rate for the route (if any)
- The devices using the travel route (if any). In R3B3, only DMS devices may be considered to be using a travel route. Any DMS that contains a traveler info message that includes the route (even if the message is not enabled) will be considered to be using the travel route.
- The roadway route(s) that are included in the travel route.
- The roadway direction(s) that are included in the travel route.
- The Counties that are included in the travel route.

A Set Columns link at the top of the page allows the user to choose which of the columns are shown / hidden on this page. When the link is clicked, a window pops up to allow the user to place a check mark next to each column name to be displayed, and to remove check marks from columns they do not wish to display (see below). The Travel Route Name and Actions columns may not be hidden and will always contain a check mark.



Column Display

- ☒ Travel Route Name
- ☒ Length
- ☒ Travel Time
- ☒ Trend
- ☒ Speed
- ☒ Toll Rate
- ☒ Used By
- ☒ Route
- ☒ Direction
- ☒ County
- ☒ Actions

Submit Cancel

Figure 5-3 Set Travel Route List Columns

When columns are hidden, a link will appear to allow the user to restore the list to display the default columns.

Each column heading is a link that when clicked causes the list to be sorted on that column. After the list is sorted on a column, the user may click the column heading link again to toggle between an ascending and descending sort.

Seven of the ten columns provide the ability to filter the list based on values in the column as follows:

- The Trav Time column provides the ability to filter the list to show only travel routes that have a travel time greater than a specified number of minutes. This filter also allows the list to be filtered to show travel routes with no travel time, and those where travel time is not applicable (no links configured).
- The Trend column provides the ability to filter the list to show only travel routes with an increasing travel time (Up), decreasing travel time (Down), or steady travel time (Flat). This filter also allows the list to be filtered to show travel routes with no travel time, and those where travel time is not applicable (no links configured).
- The Speed column provides the ability to filter the list to show only travel routes with an average speed within a certain range (0-30, 30-50, >50). This filter also allows the list to be filtered to show travel routes with no travel time, and those where travel time is not applicable (no links configured).
- The Toll Rate column provides the ability to filter the list to show only travel routes with a toll rate greater than a specified dollar amount. This filter also allows the list to be

filtered to show travel routes with no toll rate, and those where toll rate is not applicable (no toll rate source configured).

- The Route column provides the ability to filter the list to show only travel routes that include some portion of the given roadway route.
- The Dir column provides the ability to filter the list to show only travel routes that include a specific direction of roadway.
- The County column provides the ability to filter the list to show only travel routes that contain roadway within a specific county.

Links are provided for each travel route to allow the details for the route to be viewed and to allow suitably privileged users to edit the travel route or remove the travel route. A link is also available on the page to allow a travel route to be added to the system. Details on these actions are provided in the sections below.

5.1.1.2 Adding Travel Route

Administrative users can add a travel route to the system by clicking the Add Travel Route link on the travel route list. Upon clicking the link, the following form appears:

the travel time for the travel route. If no links are specified, travel times will not be supported for the travel route. See the “Travel Route Links” section below for details.

- **Toll Rate Source:** This optional selection is used to specify the source of toll rate data for the travel route. If none is specified, toll rates will not be supported for the travel route. See the “Travel Route Toll Rate Source” section below for details.
- **Location Settings:** The lists of counties and routes that the travel route traverses, as well as the total length of the travel route. If at least 1 link is specified, the user may choose to obtain the location information directly from the list of links rather than manually entering the data.
- **Travel Time Enabled:** This setting is used to enable and disable travel times for the route. Travel times are automatically disabled if the route has no links specified. This setting allows the travel times to be disabled and re-enabled without losing the current list of links or travel time related settings.
- **Max # of Links Below Quality Threshold:** The maximum number of links whose travel time data can be below the minimum quality level specified for the link. When set to zero, all links specified for the travel route must have travel time data that meets or exceeds the minimum quality level for the link. If even one link has a quality level below its minimum quality level, the travel time for the route will not be made available to other parts of the system (such as DMS). This setting can be increased to make this policy more lax for the travel route.
- **Max Travel Time (mins):** The maximum travel time allowed for the travel route. A travel time that is computed above this value will be deemed invalid and will not be made available to other parts of the system.
- **Min Travel Time (mins):** The minimum travel time allowed for the travel route. A travel time that is computed to be below this amount of time will be set to this time.
- **Alert and/or Notification Travel Time (mins):** The travel time that when met or exceeded may cause an alert and/or notification to be sent, depending on the alert and notification settings that follow.
- **Send Alert to Op Center:** When enabled, an alert will be sent to the specified op center when the travel time meets or exceeds the alert/notification travel time.
- **Send Notification to Group:** When enabled, a notification will be sent to the specified notification group when the travel time meets or exceeds the alert/notification travel time.
- **Toll Rate Enabled:** This setting is used to enable and disable toll rates for the route. Toll rates are automatically disabled if a toll rate source is not specified for the travel route. This setting allows toll rates to be disabled and re-enabled without losing the current toll rate source or toll rate related settings.
- **Toll Rate Alerts Enabled:** This setting is used to cause toll rate alerts to be sent to a specified op center when certain conditions are detected.

- **Toll Rate Notifications Enabled:** This setting is used to cause toll rate notifications to be sent to a specified notification group when certain conditions are detected.

After filling in the form and optionally adding one or more links to the travel route and/or a toll rate source, the submit button can be clicked to add the travel route to the system. The travel route list will be shown and the new travel route will appear in the list.

5.1.1.3 Travel Route Links

Links must be added to a travel route to allow travel times to be computed for the route. The links associated with a travel route can be specified when adding the route to the system via the Add Travel Route form, as part of an edit operation to edit the travel route configuration, or as a stand alone operation from the travel route details page. If no links have been specified for the travel route, an empty table of links will be shown and an Add Link option will appear, as shown below:

Links (Optional, required for Travel Time)

System	Ext ID	Name	Route	Direction	Length	County	Dist From	Prior
Add Link								

Figure 5-5 Travel Route Links - Empty List

When the “Add Link” link is clicked, the form used to select links is shown (see below).

Select Link

Search for Links

County	Road Type	Road	Direction:	Ext ID:
Any	Any	Any	Any	

Figure 5-6 Select Link, No Prior Link Selected

Because of the fairly large number of links that exist in the system for selection, the system first allows the user to enter search criteria. After picking one or more search criteria, the user must click the “Search” button to see the results of the query. The user can click the “Show All” button to see all links available in the system. Following is an example of the screen after a search is executed.

Select Link

Search for Links

County:
 Road Type:
 Road:
 Direction:
 Ext ID:

Select	Source	Ext ID	Name	Road	Direction	Length	County	Dist From Prior
<input type="checkbox"/>	INRIX	125P06000	Link Name 1	I-95	Northbound	0.500 mi	Baltimore	0.0 mi
<input type="checkbox"/>	INRIX	125P06001	Link Name 2	I-95	Northbound	0.267 mi	Baltimore	0.0 mi
<input type="checkbox"/>	INRIX	125P06002	Link Name 3	I-95	Northbound	0.345 mi	Baltimore	0.0 mi

Figure 5-7 Select Link – Search Results

The user may choose one or more of the links and click the “Add Link(s) to Travel Route” button to add the selected links to the travel route. After submitting the form, the links that were added appear in the list of links as shown below.

Links (Optional, required for Travel Time)

System	Ext ID	Name	Route	Direction	Length	County	Dist From Prior	
INRIX	125+05270	I-95 EXIT 74	I-95	Northbound	2.458 mi	Baltimore	N/A	settings move down remove
INRIX	125P05270	I-95 EXIT 74	I-95	Northbound	0.458 mi	Baltimore	0.0 mi	settings move up remove
INRIX	126P07340	I-695 EXIT 7	I-695	Inner Loop	1.458 mi	Baltimore	17.256 mi	settings move up remove

[Select Next Link](#)

Figure 5-8 Travel Route Links - Populated List

The list of links is shown in order with the “Dist From Prior” column showing the distance from the beginning of the link to the end of the link above it. When links are ordered properly, this distance will be near zero. The user may use the “move up” and “move down” links to re-order the list, and may use the “remove” link to remove a link from the list. The “settings” link can be used to set settings that apply to the link’s use within the travel route. The user may drag their mouse over the “settings” link to see the current settings. When the settings link is clicked, a form appears that allows the link percent and travel time quality to be set for the link (see below).

Link Settings for Travel Route

These settings apply to the link in the travel route where it is used.
If a link is used in multiple travel routes, it can have separate settings for each of the travel routes where it is used.

Route: I-95 NB Exit 73 to Exit 74
Link: INRIX, 125+05270, I-95 EXIT 74

Travel Time Settings

Link Percent: percent

Minimum Travel Time Quality:

Figure 5-9 Link Settings for Travel Route

The link percent specifies the portion of the link that is contained within the travel route. This is most useful for the first link within a travel route because travel routes are often defined to start at a DMS location, and the DMS might not be located at the beginning of a defined link. (Travel routes usually start at a DMS to allow travel time messages to imply that the travel time listed is from the point where the person is reading the message – i.e. the DMS). When a percentage less than 100% is specified, the travel route will use only that percent of the link's travel time when computing the overall travel time for the route. Likewise, only that percentage of the link's length will be used when computing the overall length of the travel route.

The minimum travel time quality specifies the lowest quality of travel time data that is acceptable for the link. If travel time data is received for the link that is below the specified minimum quality, the travel route may not be able to compute the overall travel time for the route, depending on the "Max # of Links Below Quality Threshold" setting for the travel route. After changing the settings for a link, the user clicks the "Submit" button to save the settings and return to the prior page (which will be the Add/Edit Travel Route form or the Travel Route Details page).

The "Select Next Link" link below the list of currently selected links can be used to add one or more links to the end of the list. When this feature is used the system can use location based information for the last link in the list to automatically provide candidates for the next link on the travel route (see below).

Select Link

Prior Link

Source	Ext ID	Name	Road	Direction	Length	County
INRIX	125P05270	I-95 EXIT 74	I-95	Northbound	0.458 mi	Baltimore

Best Candidates for Next Link(s) [Search for Others](#)

Select	Source	Ext ID	Name	Road	Direction	Length	County	Dist From Prior
<input type="checkbox"/>	INRIX	125P05270	I-95 EXIT 74	I-95	Northbound	0.458 mi	Baltimore	0.0 mi
<input type="checkbox"/>	INRIX	125P05270	I-95 EXIT 74	I-95	Northbound	0.458 mi	Baltimore	0.0 mi
<input type="checkbox"/>	INRIX	125P05270	I-95 EXIT 74	I-95	Northbound	0.458 mi	Baltimore	0.0 mi
<input type="checkbox"/>	INRIX	125P05270	I-95 EXIT 74	I-95	Northbound	0.458 mi	Baltimore	0.0 mi
<input type="checkbox"/>	INRIX	125P05270	I-95 EXIT 74	I-95	Northbound	0.458 mi	Baltimore	0.0 mi

Add Link(s) to Travel Route

Cancel

Figure 5-10 Select Link - Prior Link Exists

The user may choose one or more of the candidate links, or can click “Search for Others” to perform a search for other links as was required when the list of selected links was empty (no prior link). The screen capture below shows the form when “Search for Others” has been clicked.

Select Link

Prior Link

Source	Ext ID	Name	Road	Direction	Length	County
INRIX	125P05270	I-95 EXIT 74	I-95	Northbound	0.458 mi	Baltimore

Search for Links [Show Best Candidates](#)

County	Road Type	Road	Direction:	Ext ID:
Any	Any	Any	Any	

Search

Show All

Add Link(s) to Travel Route

Cancel

Figure 5-11 Select Link - Search for Links

5.1.1.4 Travel Route Toll Rate Source

A Toll Rate Source may be specified for a travel route to allow toll rates to be included in the travel route. If a toll rate source is not specified, the route will not support toll rates. The toll rate source may be specified while adding a travel route to the system, editing an existing travel route's configuration, or from the travel route's details page. A table showing the current toll rate source (if any) will appear on the form or details page as shown below.

Toll Rate Source (Optional, required for Toll Rate)

System	Start ID	End ID	Description
Add Toll Rate Source			

Figure 5-12 Toll Rate Source Unspecified

When the “Add Toll Rate Source” link is clicked, a form appears to allow the user to select the toll rate source for the travel route (see below).

Select Toll Rate Source

Select	System	Start ID	End ID	Description
<input type="radio"/>	Vector	521	538	ICC DMS 101 to Exit 1
<input type="radio"/>	Vector	522	539	ICC DMS 102 to Exit 2
<input type="radio"/>	Vector	523	540	ICC DMS 103 to Exit 3
<input type="radio"/>	Vector	524	541	ICC DMS 104 to Exit 4
<input type="radio"/>	Vector	525	542	ICC DMS 105 to Exit 5
<input type="radio"/>	Vector	526	543	ICC DMS 106 to Exit 6
<input type="radio"/>	Vector	527	544	ICC DMS 107 to Exit 7

Figure 5-13 Select Toll Rate Source

The user will select the toll rate source by clicking a radio button on the left corresponding to the toll rate source they wish to select and clicking the “Select Toll Rate Source” button. Their selection will be saved and the Toll Rate Source section of the prior page will show their selection.

Toll Rate Source (Optional, required for Toll Rate)

System	Start ID	End ID	Description	
Vector	546	834	ICC DMS 123 to Exit 7	Change Remove

Figure 5-14 Toll Rate Source Specified

The “Change” link can be used to select a different toll rate source for the route, and the “Remove” link can be used to remove the toll rate source from the route (effectively disabling toll rates for the travel route).

5.1.1.5 Edit Travel Route

Editing a travel route is similar to adding a travel route and uses the same form that is used when adding a travel route, however the form is pre-populated with the travel route's existing configuration data when editing. See Adding Travel Route above for details.

5.1.1.6 Remove Travel Route

A travel route may be removed using the "Remove" link on the travel route list page. A warning message will confirm the user's intention and also indicate if the travel route is used by other objects in the system. After the user removes a travel route, the travel route list will be refreshed and the travel route will no longer appear.

5.1.1.7 Travel Route Details

The details page for a travel route can be accessed by clicking the "details" link for the travel route on the travel route list page. The travel route details page shows status information for the travel route in addition to the travel route's configuration settings. The data is organized into sections on the details page – each section is shown and discussed below:

Status
Travel Time: 10 mins (as of 14:10)
Trend: Up (as of 14:10)
Speed: 55 mph (as of 14:10)
Toll Rate: \$3.25 (at 14:05)
Used By:  [XYZ](#)

Figure 5-15 Travel Route Details - Status Section

The Status section shows the current status of the travel route. The Travel Time will be shown if the travel route has one or more links defined and each has travel time data. If the travel time is not valid due to quality, age, etc. the travel time will still be shown, however it will be grayed out and a message will state that the time is not valid and the reason. The time next to the travel time is the time the travel time was computed. The Trend is the travel time trend, and only applies if travel time applies. The Speed is computed based on the travel route length and the current travel time and only applies if travel time applies. The Toll Rate is the toll rate supplied by the toll rate source, if any, and the toll rate's effective time. The Used By field shows a list of any CHART objects that are configured to utilize the data from the travel route. For R3B3 only DMSs use data from travel routes. Clicking on the name of an object listed in the Used By field will cause that object's details page to be shown.

Link Status				
Ext ID	Name	TT	Trnd	Speed
125+05270	I-95 EXIT 74	3 mins	Up	55
125P05270	I-95 EXIT 74	4 mins	Up	47

Figure 5-16 Travel Route Details - Link Status Section

The Link Status section of the travel route details page (shown above) shows the travel time, trend, and speed for each link included in the travel route (if any). The Ext ID column is the identifier for the link in the external system that supplies travel time data for the link (INRIX for R3B3). The TT column contains the current travel time, Trnd has the current trend, and the Speed column contains the computed speed for the link based on its travel time and length.

Link Travel Time History

ID	Name	14:10	14:05	14:00	13:55	13:50	13:45	13:40	13:35	13:30	13:25	13:20	13:15
125+05270	I-95 EXIT 74 (Quality)	3 mins (high)	3 mins (high)	3 mins (high)	2 mins (high)	2 mins (high)	2 mins (high)	2 mins (high)	2 mins (high)	2 mins (high)	2 mins (high)	2 mins (high)	2 mins (high)
125P05270	I-95 EXIT 74 (Quality)	4 mins (high)	4 mins (high)	4 mins (high)	4 mins (high)	4 mins (high)	3 mins (high)	3 mins (high)	3 mins (high)	3 mins (high)	3 mins (high)	3 mins (high)	3 mins (high)
TOTAL		7 mins	7 mins	7 mins	6 mins	6 mins	5 mins	5 mins	5 mins	5 mins	5 mins	5 mins	5 mins

Figure 5-17 Travel Route Details - Link History Section

The Link Travel Time History section of the travel route details page (see above) shows the recent travel time readings received for each of the travel route's links (if any). The travel times are organized into 5 minute increments starting from the current 5 minute period. The travel time quality is shown in addition to each link's travel time. The Total row contains the travel time computed for the travel route. One or more intervals could be empty if data was not received for a link during that period. If two or more travel time readings are received for a link within the same interval, the latest will be used.

Toll Rate History

Toll Rate Src.	14:10	14:05	14:00	13:55	13:50	13:45	13:40	13:35	13:30	13:25	13:20	13:15
ICC DMS 123 to Exit 7	\$3.25	\$3.25	\$3.25	\$3.25	\$3.25	\$3.25	\$3.25	\$2.25	\$2.25	\$2.25	\$2.25	\$2.25

Figure 5-18 Travel Route Details - Toll Rate History Section

The Toll Rate History section of the travel route details page (shown above) shows the recent history of toll rate data received from the travel route's toll rate source (if any). The toll rate data is organized into 5 minute increments starting with the current 5 minute time period. One or more intervals could be empty if data is not received during that interval. If two or more toll rate updates are received for the toll rate source within the same interval, the latest will be shown.

Link Configuration

System	Ext ID	Name	Road Direction	Length	County	Dist From Prior	
INRIX	125+05270	I-95 EXIT 74 I-95	Northbound	2.458 mi	Baltimore	N/A	settings move down remove
INRIX	125P05270	I-95 EXIT 74 I-95	Northbound	0.458 mi	Baltimore	0.0 mi	settings move up remove

[Select Next Link](#)

Figure 5-19 Travel Route Details - Link Configuration Section

The Link Configuration section of the travel route details page (see above) shows the configuration settings for the links included in the travel route (if any). Features exist to allow the settings for each link to be changed, the order of the links to be changed, or to add/remove links. Each External Link ID is a browser link that causes the details page for the link to be shown.

Toll Rate Configuration				
System	Start ID	End ID	Description	
Vector	546	843	ICC DMS 123 to Exit 7	Change Remove

Figure 5-20 Travel Route Details - Toll Rate Configuration Section

The Toll Rate Configuration section of the travel route details page shows the currently selected toll rate source (if any). Links exist to allow a new/different toll rate source to be added and to allow the currently selected toll rate source to be removed.

[General Settings](#) [\(Edit\)](#)
Destination (preferred): White Marsh
Destination (option 1): Wht Mrsh
Destination (option 2): N/A

Figure 5-21 Travel Route Details - General Settings Section

The General Settings section of the travel route details page (shown above) shows the current general settings and provides a link to allow the settings to be edited. The preferred destination name for the travel route and two shorter alternatives are included.

[Location Settings](#) [\(Edit\)](#)
Source: Derived from Links
Counties: Baltimore
Route: I-95
Direction(s): North
Length: 4.374 mi

Figure 5-22 Travel Route Details - Location Settings Section

The Location Settings section of the travel route details page shows the location related fields. The Source field indicates if the location is based on data from the links included in the travel route (if any), or manually entered by a user. A link exists to allow the location settings to be edited.

[Travel Time Settings](#) [\(Edit\)](#)
Travel Time Enabled: Yes
Max Displayable Travel Time (mins): 25
Min Travel Time (mins): 5
Max # of Links Below Quality Threshold: 0
Alert/Notification Travel Time (mins): 25
Travel Time Alerts Enabled: Yes
Op Ctr To Alert: SOC
Travel Time Notifications Enabled: Yes
Group To Notify: Travel Time Group

Figure 5-23 Travel Route Details - Travel Time Settings Section

The Travel Time Settings section of the travel route details page (shown above) shows the current value for each of the travel time related settings. An Edit link provides the ability for users with proper rights to edit any of the settings. These settings are described in the Adding Travel Route section 5.1.1.2 above.

<u>Toll Rate Settings</u> (Edit)	
Toll Rate Enabled:	Yes
Toll Rate Alerts Enabled:	Yes
Op Ctr To Alert:	SOC
Toll Rate Notifications Enabled:	Yes
Group To Notify:	Toll Rate Group

Figure 5-24 Travel Route Details - Toll Rate Settings Section

The Toll Rate Settings section of the travel route details page (see above) shows the values of settings related to toll rates. An edit link allows users with proper rights to edit these settings.

5.1.1.8 Link Details

The Link Details page is accessible by clicking the External ID of a link shown on the travel route details page.

Link Details: I-95 EXIT 74

Status

Travel Time: 4 mins

Status Time: 14:10

Trend: Up

Speed: 55

History

	14:10	14:05	14:00	13:55	13:50	13:45	13:40	13:35	13:30	13:25
Travel Time (Quality)	3 mins (high)	3 mins (high)	3 mins (high)	2 mins (high)	2 mins (high)	2 mins (high)	2 mins (high)	2 mins (high)	2 mins (high)	2 mins (high)

Configuration

System: INRIX

Ext ID: 125+05270

Link Name: I-95 EXIT 74

Counties: Baltimore County

Route Type: I (Interstate)

Route: I-95

Direction: Southbound

Link Length (mi): 0.425

Starting Latitude: 38.0012

Starting Longitude: -77.0012

Ending Latitude: 38.0024

Ending Longitude: -77.0024

Back

Figure 5-25 Link Details

The current status of the link is shown along with its travel time history. Configuration data for the link as obtained from the external system is displayed but is not editable by any user.

5.1.2 DMS Message Templates

DMS Message Templates are used to define the format and layout of messages used to show travel times and/or toll rates on DMSs. Message templates target DMSs of a specific size (rows / columns) to allow layout features such as columnular data and to ensure messages created with the template match the layout specified by the template designer. Templates may contain both text and data fields. Data fields will be replaced with actual data from one or more Travel Routes when a template is used for a DMS message. The format of each type of data field can be set per message template, and field group numbers are used to indicate that two or more fields in a message are to obtain their data from the exact same data source. R3B3 provides features to view existing message templates, add new templates, edit existing templates, and remove templates. These features are discussed further in the sections below.

5.1.2.1 View DMS Message Templates

The list of DMS Message Templates can be viewed using the View/Edit link on the system profile page in the Travel Time / Toll Rate Settings section (see below).

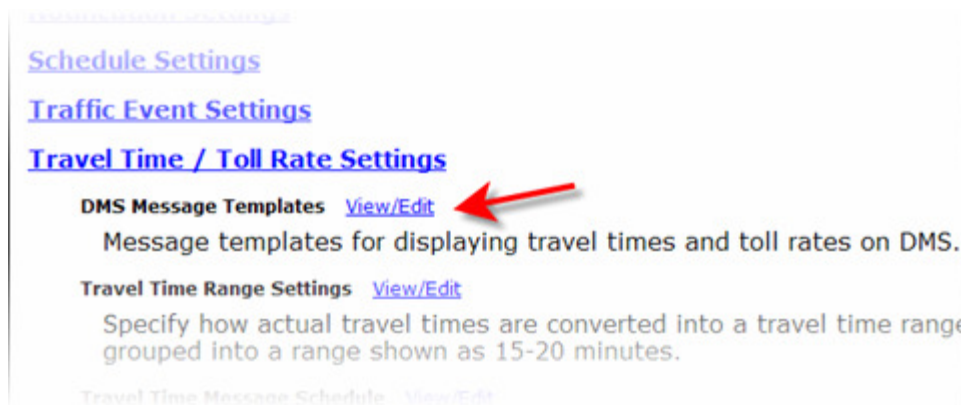


Figure 5-26 View / Edit DMS Message Templates Link

After clicking the View/Edit link, the Travel Time / Toll Message Templates page is shown listing all existing templates.

Travel Time / Toll Message Templates									
Set Columns (show default columns)									
Template	Sign Size	Page 1	Page 2	Example Message	Actions	Travel Time Format	Travel Time Range Format	Toll Rate Format	Toll Time Format
TT to DEST multiple routes 3x21	3 x 21	<TT1> TO <DEST1> <TT2> TO <DEST2> <TT3> TO <DEST3>	<TT4> TO <DEST4> <TT5> TO <DEST5> <TT6> TO <DEST6>	999 MIN TO 999 MIN 999 MIN TO 999 MIN 999 MIN TO 999 MIN	Edit Remove	999 MIN			
TR to DEST and TT 1 route 3x18	3 x 18	<TR1> TO <DEST1> <TT1>		\$9.99 TO 999 MIN 999 MIN	Edit Remove	999 MIN		\$9.99	
ICC Toll Msg 3x8	3 x 8	<TR1> <TR2> <TRTIME>		\$9.99 12:59 PM	Edit Remove			\$9.99	12:59 PM

New Template: -- Select Size --

Back To System Profile

Figure 5-27 Travel Time / Toll Message Template List

The following columns are shown with data for each template as applicable:

- **Template** – The name of the template assigned by the user.
- **Sign Size** – The size of the sign for which the template can be used (rows / columns)
- **Page 1** – A text representation of the first page of the template.
- **Page 2** – A text representation of the second page of the template.
- **Example Message** – An example of the message layout and format, using place holders for data fields.
- **Actions** – Links that allow the template to be edited or removed.
- **Travel Time Format** – The format used for travel time fields included in the template (if any)
- **Travel Time Range Format** – The format used for travel time range fields included in the template (if any).
- **Toll Rate Format**– The format used for toll rate fields included in the template (if any).
- **Toll Time Format** - The format used for toll rate time fields included in the template (if any)
- **Route Length Format** – The format used for route length fields included in the template (if any).

The template list can be sorted on the following columns: template name, sign size, travel time format, travel time range format, toll rate format, toll time format, and route length format.

The template list can be filtered based on values in the following columns: sign size, travel time format, travel time range format, toll rate format, toll time format, and route length format.

The template list also supports showing / hiding columns. The Set Columns link can be clicked to show a list of the columns with a check mark next to each column that is currently displayed. The user can uncheck columns they wish to hide, or check columns they wish to show and submit the form. The template list will be refreshed to reflect the user's settings.

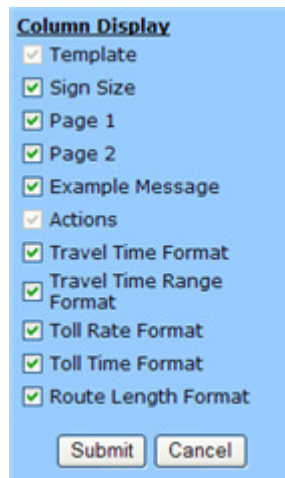


Figure 5-28 Template List Column Settings

If the column display settings have been changed from the default, a “Show Default Columns” link will appear that allows the column settings to be returned to the default. Column settings are saved on a per user / per machine basis, meaning if the user logs out of CHART and logs back in the future using the same machine user account, browser, and CHART user account, their prior column display settings will still be in effect.

The New Template select list at the bottom of the template list allows the user to create a new template for a sign size they select. See the section below for more information regarding adding DMS Message Templates.

5.1.2.2 Add DMS Message Template

Message templates are added to the template list using the New Template select list that appears below the list of templates. To add a template, the user first selects a DMS size (rows x columns) for which the template will be used.

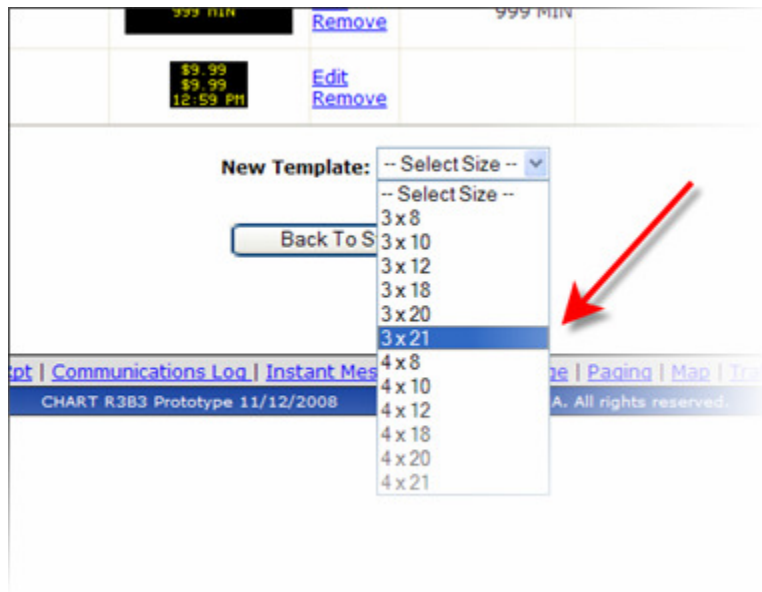


Figure 5-29 Add Message Template - Select Sign Size

After selecting the sign size, the Add Template form will appear, shown below. The user can add text and/or data fields to any row of the message template. As entries are made in the rows, the display at the top of the page shows a graphical representation of a message specified by the current template settings, using “dummy data” for data fields. Data fields are added as “tags” that represent the type of field and the data source index number. All tags that have the same index will be replaced with data from the same data source when the template is used to create an actual message. Data fields in DMS messages that are created using a template will be updated as the data changes in the data source(s) used for the message.

CHART - Windows Internet Explorer

CHART Main Window Help

Edit Travel Time / Toll Message Template: 3 X 21

TRAVEL TIME
TO *** XX-YY MIN**
TO *** XX-YY MIN**

Page 1

Row 1:	<input type="radio"/> Left <input checked="" type="radio"/> Center <input type="radio"/> Right	<input type="text" value="Travel time"/> Tag
Row 2:	<input type="radio"/> Left <input checked="" type="radio"/> Center <input type="radio"/> Right	<input type="text" value="To <DEST1> <TTRANGE1>"/> Tag
Row 3:	<input type="radio"/> Left <input checked="" type="radio"/> Center <input type="radio"/> Right	<input type="text" value="To <DEST2> <TTRANGE2>"/> Tag

Page 2

Row 1:	<input type="radio"/> Left <input checked="" type="radio"/> Center <input type="radio"/> Right	<input type="text"/> Tag
Row 2:	<input type="radio"/> Left <input checked="" type="radio"/> Center <input type="radio"/> Right	<input type="text"/> Tag
Row 3:	<input type="radio"/> Left <input checked="" type="radio"/> Center <input type="radio"/> Right	<input type="text"/> Tag

Travel Time Range Format:

Justify within DEST tag:

If Route Data Missing: ☐ Discard Message ☒ Discard Page ☐ Discard Row

Page On Time (Seconds): **Page Off Time (Seconds):**

Description

To limit the size of a DEST tag, enter the number of characters in front of "DEST". For example, <12DEST1> will occupy 12 characters.

CHART R3B3 Prototype 11/12/2008 © 2002-2008 MDSHA. All rights reserved.

Figure 5-30 Add DMS Message Template

In the example above, Row 2 of the first page has a <DEST1> tag to include the destination from a travel route, followed by a <TTRANGE1> tag to include the travel time range from that same data source. (The index “1” in both the DEST1 and TTRANGE1 tags indicate both pieces of data will be obtained from the same data source.) Adding a tag to the message is done by clicking the “Tag” link next to the row where the tag will be added. This causes a popup to appear that provides links used to add any of the available tags to the row (see below).

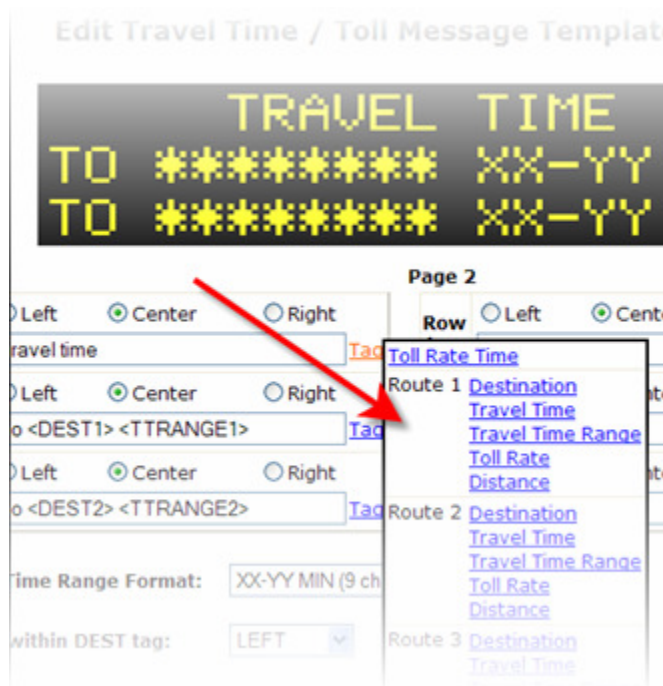


Figure 5-31 DMS Message Templates - Adding Data Fields

The following data fields are available for use in a template:

- Destination** – Destination fields will be replaced with the largest of the three destinations specified for a travel route that fits in the space allotted for the field (preferred, option 1, or option2). Destination fields use all available space by default. A number can be entered at the beginning of the field tag (such as <6DEST1>) to make the destination field use a specific width. This feature can be useful when aligning data in columns. Destination fields are represented as asterisks in the graphical display when using the default width, and as X's when set to use a specific width. When at least 1 destination tag is used in a template, a Justify Within DEST Tag field will appear on the form to allow the user to specify if destinations that do not use the entire width of the dest field will be left, right, or center justified.
- Travel Time** – Travel time fields (represented by a tag of the form <TT1>) will be replaced with a travel route's current travel time. After at least 1 travel time field is included in the template, a Travel Time Format field will appear on the form to allow the user to specify the format to be used for all travel time fields in the template. The format chosen ultimately specifies the width of the field in addition to the time format.
- Travel Time Range** – Travel Time Range fields (represented by a tag of the form <TTRANGE1>) will be replaced by a travel time range rather than the actual travel time. A travel time range is created by taking the actual travel time from a travel route and subtracting a few minutes to get a low range, and adding a few minutes to get a high range, yielding a range such as 11-14 minutes. The number of minutes added/subtracted are specified in the system profile, and can differ based on amount of the actual travel time. Like the travel time field, once a travel time range field is used in the message

template a format field will appear on the form that allows the user to choose the format used for all travel time range fields in the template. Each format specifies the width required for the field in addition to the number of digits for the travel times, whether or not “Minutes” or some abbreviation of minutes will appear, etc.

- **Toll Rate** – Toll Rate fields (represented by a tag of the form <TR1>) will be replaced with the current toll rate for a travel route. After at least one toll rate tag is included in the template, a Toll Rate Format field will appear on the form to allow the user to choose the format used for all toll rate fields that appear in the template. Like the other field formats, each toll rate format also specifies the width of the field.
- **Distance** – Distance fields (represented by a tag of the form <DISTANCE1>) will be replaced by the distance specified for a travel route. This distance may be derived from the links included in the travel route or may have been manually entered by a user. When at least 1 distance field is used in the template, a Distance Format field will appear on the form to allow the user to specify the format to use for all distance fields in the template. Distance formats specify the size of the field in addition to the number of digits, whether tenths of a mile are included, and whether or not the word “miles” or an abbreviation of “miles” is included in the field.
- **Toll Rate Time** - Toll Rate Time Fields (represented by a tag of the form <TRTIME>) will be replaced by the latest toll rate effective time for a toll rate included in the template. Note that toll rate time does not contain an index like the other fields – the toll rate time used is determined by the system. When at least one toll rate time field is included in the template, the Toll Rate Time Format field will appear to allow the user to select the format used for all toll rate time fields in the template. The toll rate time format specifies the time format to use which ultimately specifies the width of the field.

By default, each row of the template will be centered, however the user can also choose to left or right justify on a row by row basis. The user may choose the “Page On Time” to control how long each page of a two page message will be displayed before blanking the sign and then displaying the next page. The “Page Off Time” setting allows the user to choose how long the sign will remain blank when switching pages.

The “If Route Data Missing” setting specifies how the template is to be used if data for a field included in the template is missing for any reason. The user can choose to discard the entire message, discard the page of the message with missing data, or discard the row with missing data. The best choice for this setting will depend on the content of the message template. For example, if the template is set up such that travel times for multiple destinations are listed, it may be OK to just discard a row if data is missing. If, however, the template has data for a single route spread across multiple rows of the DMS, it is probably better to discard the entire page or message.

The user can enter a description of the template which will appear in the template list as well as the select list used to choose a template for a DMS message.

Normally, the advanced form of the template editor is shown, but it is possible for the user to show a simpler version of the editor by clicking the “Show Def. Editor” button. The default

version of the editor does not allow the user to choose page justification or page on/off times. Other than that, the default editor works the same way as the advanced editor (see below).

Figure 5-32 DMS Message Templates - Default Editor

In both the advanced and default versions of the editor, a Check Spelling button appears that allows the user to perform a spelling check on the message.

5.1.2.3 Edit DMS Message Template

Existing templates can be edited by clicking the “Edit” link for a template in the template list. This will cause a form identical to the form shown above for adding a template to appear pre-populated with the template’s data. The user can then make any changes to the template and save their changes.

5.1.2.4 Remove DMS Message Template

Existing templates can be removed from the system by clicking the “Remove” link for a template in the template list. A warning message will confirm the user’s intent before removing the template from the system.

5.1.3 Travel Time and Toll Rate Messages

Travel time and toll rate messages are created by combining a message template with data from one or more travel routes. Whether a message will contain travel time(s), toll rate(s), or both depends on the data fields in the template and the data available for the selected travel route(s). Travel Time and Toll Rate messages can be configured for any DMS as long as there is a DMS message template available that matches the size of the DMS. A DMS may have multiple travel time / toll rate messages configured for use on the DMS, however only one may be enabled at any given time. A list of travel time / toll rate messages configured for a DMS is shown on each DMS details page. A user with appropriate rights can add, edit, remove, enable, and disable travel time messages. See the sections below for details.

5.1.3.1 Travel Time / Toll Rate Message List

The travel time / toll rate message list appears on the details page of each DMS (see below). It shows the currently configured messages (if any), the status of each message (enabled or not), and provides links to allow each message to be enabled/disabled, edited, or removed. A link is also provided to allow a new travel time / toll rate message to be added to the DMS configuration.

Travel Time / Toll Message (Add)		
Message	Status	Action
	Enabled	Disable Edit Remove
		Enable Edit Remove

Figure 5-33 Travel Time / Toll Rate Message List on DMS Details Page

The graphical representation of each message shown in this list uses the actual data from the travel routes specified in the message when available. When actual travel route data is not available, “dummy data” will be used in the data fields in the message.

5.1.3.2 Add Travel Time / Toll Rate Message

When the user chooses to add a travel time / toll rate message to a DMS, a form is shown to allow the user to specify the message template to be used for the message and the travel routes to be used to supply data to the data fields in the template (see below).

CHART - Windows Internet Explorer

CHART [Main Window](#) [Help](#)

Select Travel Time / Toll Message for BigTS3001

Message Template:
 -- Select --

Page 1	Page 2

Travel/Toll Routes:
 N/A

Formatted Message:
 N/A

☐ Automatic row positioning

OK Cancel

Figure 5-34 Add Travel Time / Toll Rate Message - Initial Form

The Message Template list will contain all templates that have a size that matches the size of the DMS. After the user selects a template, the content of the template is shown and a select field will appear for each different data source specified in the template as shown below.

CHART - Windows Internet Explorer

CHART Main Window Help

Select Travel Time / Toll Message for BigTS3001

Message Template:
 TT to DEST multiple routes ▼

Page 1	Page 2
<TT1> TO <DEST1>	<TT4> TO <DEST4>
<TT2> TO <DEST2>	<TT5> TO <DEST5>
<TT3> TO <DEST3>	<TT6> TO <DEST6>

Travel/Toll Routes:

Route 1 NONE ▼

Route 2 NONE ▼

Route 3 NONE ▼

Route 4 NONE ▼

Route 5 NONE ▼

Route 6 NONE ▼

Formatted Message:

N/A

☐ Automatic row positioning

OK Cancel

Figure 5-35 Add Travel Time / Toll Rate Message - Template Selected

Each Route selection list corresponds to the data fields in the template with the same index as the Route list. The user selects the travel route that will be the source of data for those fields. For example, the list labeled “Route 2” corresponds to all data fields whose tag ends with the number “2”. In the example above, that would be the <TT2> and <DEST2> fields. The travel route selected for Route 2 will be used to supply data for those fields.

The user is not required to select a route for each data field index that exists in the template. If a travel route is not selected for an index that appears in the template, the fields with that index will be considered to have missing data, and the display of the message will depend on the template’s missing data setting.

As the user selects routes, the Formatted Message section of the form will show a graphical display of the message that will result given the current template selection and route selection(s).

The Automatic row positioning feature can be selected to cause the system to use an automatic vertical layout when all rows of a message page are not filled (either due to the layout of the template or due to missing data when the “discard row” missing data rule exists in the template).

The automatic vertical layout rules cause row 2 of a DMS to be used if the page has only a single row, and rows 1 and 3 to be used if the message has only 2 rows. This applies to DMSs that have 3 or 4 rows.

After the user has selected a template and selected routes to be used to supply data to the fields of the template, the user may click the OK button to save the message. The message will appear in the DMS's travel time / toll rate message list and will be disabled until specifically enabled by the user.

5.1.3.3 Edit Travel Time / Toll Rate Message

A user with sufficient rights can edit an existing travel time / toll rate message by clicking the "Edit" link next to the message in the message list. An Edit form that is identical to the Add form shown above will appear pre-populated with the user's prior selections. The user may then make changes and click the OK button to save their changes.

5.1.3.4 Remove Travel Time / Toll Rate Message

A user with sufficient rights can remove a travel time / toll rate message. A warning will be shown to prevent accidental removal. After the user confirms their intention to remove the message, the message will be disabled (if currently enabled) and will be removed from the DMS's travel time / toll rate message list.

5.1.3.5 Enable Travel Time / Toll Rate Message

A user with sufficient rights can enable a travel time / toll rate message by clicking the "Enable" link for the message in the DMS's travel time / toll rate message list. When a message is enabled, any currently enabled message will first be disabled (only one travel time / toll rate message can be enabled at a time). The travel time / toll rate message will then be placed on the DMS's arbitration queue. If the message contains at least 1 toll rate field, the message will be placed in the arbitration queue "bucket" specified in the DMS configuration for use with toll rate messages. By default, the "toll rate" bucket is used, however any of the buckets can be set for toll rate message use in the DMS configuration, including the highest priority bucket "urgent". If the message does not contain any toll rate fields, the message will be placed in the arbitration queue "bucket" specified in the DMS configuration for use with travel time messages. By default the "travel time" bucket will be used but any other bucket can be set for travel time message use in the DMS configuration. The decision of which bucket to configure for use with travel time / toll rate messages depends largely on the purpose of the DMS, and the default settings will usually be sufficient for DMSs that display traffic event and travel time/toll rate messages.



Figure 5-36 DMS Arbitration Queue With Toll Rate Message

Once placed on the DMS arbitration queue, the actual display of the travel time / toll rate message on the DMS field device is governed by the arbitration queue as well as several other factors that apply only to travel time / toll rate messages. The existing arbitration queue processing which chooses the message to display based on priority and allows single page messages to be combined (based on combinability configuration settings) applies to travel time / toll rate messages. Travel time / toll rate messages on the arbitration queue can be moved just like any other message that exists on the arbitration queue (such as those from traffic events). Removal of a travel time / toll rate message from the arbitration queue is the same as disabling the message.

Several factors specific to travel time / toll rate messages can prevent the display of the message even if the arbitration queue would otherwise display it based on priority and/or message combination rules, as follows:

- **Missing Data** – If valid data is not available for one or more of the data fields specified in the template used by a travel time / toll rate message, the ability to display the message will be based on the missing data rule specified in the template. If the rule is to discard the message, the message will not be displayed. If the rule is to discard the page, and no

other page without missing data exists, the message will not be displayed. If the rule is to discard the row, and there are no rows that do not have missing data, the message will not be displayed. Note that in addition to a broken data feed for a travel route's links or toll rate source, several other conditions may cause a travel route's data to be unavailable. This includes conditions such as travel time quality below the accepted level, travel time above the configured threshold for a travel route, disabled travel time or toll rate for a route, expired data, and others.

- **Travel Time Schedule** – If a message is considered a travel time message (due to its lack of any toll rate fields in the message), the message is subject to the travel time message schedule specified for the DMS. By default, a DMS will be configured to utilize the system-wide travel time display schedule, however the display schedule may be overridden for a DMS. The DMS may be set to allow the display of travel time messages without restriction (24x7) or may specify time periods during the day when travel time messages may be displayed, regardless of the system-wide schedule. When a travel time message is not displayed due to the travel time message display schedule, the message remains enabled and remains on the arbitration queue, ready to be displayed when the schedule allows.

5.1.3.6 Disable Travel Time / Toll Rate Message

A travel time / toll rate message can be disabled by clicking the “disable” link for the currently enabled travel time / toll rate message in the DMS's travel time / toll rate message list. Disabling the travel time / toll rate message will cause it to be removed from the DMS's arbitration queue and the arbitration queue will re-evaluate to determine if it needs to change the DMS display.

5.1.4 DMS Travel Time / Toll Rate Settings

There are several groups of settings within the DMS configuration that apply to travel time / toll rate messages: arbitration queue level settings for travel time and toll rate messages, travel time message schedule settings, and associated travel routes. Details are provided in the sections below.

5.1.4.1 DMS Arbitration Queue Level Settings

Two fields in the DMS basic configuration specify the arbitration queue “buckets” to be used for travel time and toll rate messages.

Basic Settings: [\(Edit\)](#)

Name:	4403
Network Connection Site:	fiona
Has Beacons:	YES
Sign Type:	Character Matrix
Owning Organization:	SHA
Device Logging:	OFF
Display Size:	3 X 21
Max Pages:	2
Char Size (pixels):	7 X 5 (H X W)
Font:	Font 3
Line Spacing:	1
Default Line Justification:	Center
Default Page Timing:	2.5 seconds ON, 0.0 seconds OFF
Travel Time Msg Queue Level:	Travel Time
Toll Rate Msg Queue Level:	Toll Rate




Figure 5-37 DMS Travel Time / Toll Rate Arb Queue Levels

These settings specify which of the arbitration queue “buckets” a travel time or toll rate message is placed in when the message is enabled. Once in the arbitration queue the user can move the message to a different bucket ... this setting is the bucket used for initial placement into the arbitration queue.

5.1.4.2 DMS Travel Time Message Schedule

Each DMS contains settings for its travel time message display schedule. The settings can specify that the DMS is to use the system-wide travel time display schedule or a custom schedule for the DMS. The custom schedule can specify that there are no restrictions on displaying travel time messages on the DMS (allowed 24x7), or the schedule can specify periods during the day when travel time messages are allowed.

Travel Time Message Schedule: [\(Edit\)](#)

Custom Schedule In Use	
Travel Time Message Allowed From	05:30 Until: 09:30
Travel Time Message Allowed From	15:00 Until: 18:30

Figure 5-38 DMS Travel Time Message Schedule

The current schedule settings are shown on the DMS details page (shown above). Users with rights to configure the DMS can use the “Edit” link to edit these settings. When the edit link is clicked, a form appears allowing the schedule settings to be changed (see below).

Figure 5-39 DMS Travel Time Message Schedule Form

If the user chooses Use System Default Schedule, no other settings will appear on this form and the DMS will use the system-wide travel time message schedule. When Specify Custom Schedule is chosen (as shown above), the user can choose to enable messages 24x7 (no restrictions) or choose to enable messages during specific times of the day. When enabling messages during specific times of day, the user enters one or more start and end times. The “Add Time Range” link allows more time ranges to be added. Note that the schedule applies to all days, even weekends and holidays.

5.1.4.3 DMS Associated Travel Routes

The list of travel routes that may be used in travel time / toll rate messages for a DMS may be specified for each DMS. Only travel routes previously associated with a DMS will be available for use in travel time / toll rate messages for that DMS. This requirement has two purposes:

- It keeps the route selection list (shown when creating a travel time / toll rate message) short
- It allows an administrator to specify the travel routes that are applicable to a DMS, preventing the creation of travel time/toll rate messages that are inappropriate for the DMS. For example, setting the list of applicable routes properly can prevent a message for a travel route that does not begin at the DMS from being displayed on the DMS

Available Travel Routes: [\(Edit\)](#)

[I-95 @ MD 100 to I-270](#)
[I-95 @ MD 100 to WW Bridge](#)
[I-95 @ MD 100 to MD 97 via MD 200](#)
[I-95 @ MD 100 to MD 650 via MD 200](#)
[I-95 @ MD 100 to I-270 via MD 200](#)

Figure 5-40 DMS Associated Travel Routes - DMS Details Page

The current list of travel routes associated with the DMS (if any) are shown on the DMS details page (see above). Each travel route name shown is a link to the details page for the travel route. Users with rights to configure the DMS can click the “Edit” link to edit the list of travel routes associated with the DMS.

Figure 5-41 DMS Associated Travel Routes - Edit Form

All routes available in the system that are not already associated with the DMS will be shown in the Available Routes list on the left. The user can select available routes and use the Add button to associate the routes with the DMS, or choose a previously selected route and click the Remove button to disassociate them from the DMS. When the list of Selected Routes is set as desired, the user clicks the Submit button to save their changes.

5.1.5 System Profile Travel Time Settings

There are several groups of settings in the system profile related to travel times. This includes configuration settings that specify how travel time ranges are created given an actual travel time,

the system-wide travel time message display schedule, and miscellaneous travel time settings. These settings are all accessed via links in the Travel Time / Toll Rate Settings section of the system profile page as shown below.

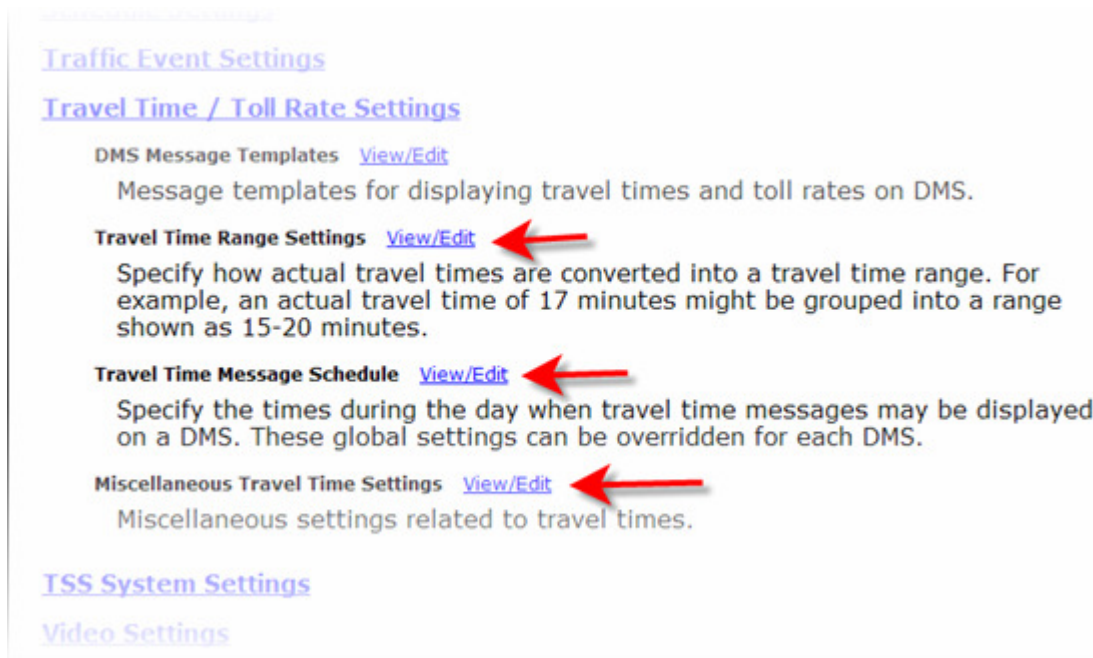


Figure 5-42 Travel Time System Profile Settings

Details on each of these groups of settings can be found in the sections below.

5.1.5.1 Travel Time Range Settings

The travel time range settings specify how the system is to create a travel time range given a single travel time.

Configure Travel Time Range Settings

When displaying a travel time range, the system takes the actual travel time and subtracts a number to get the lower range value, and adds a number to get the upper range value. The settings below allow you to set these numbers and to use different numbers based on the travel time.

Actual Travel Time Greater Than	Subtract	Add	Examples
<input type="text" value="1"/>	<input type="text" value="2"/>	<input type="text" value="2"/>	Actual: 5; Range: 3-7
<input type="text" value="10"/>	<input type="text" value="3"/>	<input type="text" value="3"/>	Actual: 20; Range: 17-23
<input type="text" value="30"/>	<input type="text" value="5"/>	<input type="text" value="5"/>	Actual: 32; Range: 27-37

[add more rows](#)

CHART R3B3 Prototype 11/12/2008 © 2002-2008 MDSHA. All rights reserved.

Figure 5-43 Configure Travel Time Range Settings

Travel time ranges are formed by starting with an actual travel time and subtracting a number from it to get the low end of the range, and adding a number to it to get the high end of the range. The system allows the numbers that are added and subtracted from the actual travel time to be specified differently based on the travel time value. The left hand column specifies the lowest travel time that uses the add/subtract numbers on that row. The first row always has 1 as the left hand column. To use a single set of add/subtract values for ALL travel times, only that first row is needed. All travel times greater than 1 will use the add/subtract numbers in that row. If, however, different add/subtract numbers are to be used as the travel time becomes larger, additional rows can be added to specify the thresholds at which the add/subtract numbers change. In the example above, the range created for all travel times from 1 to 9 will subtract 2 and add 2 to the actual travel time. For travel times from 10 to 29, the add/subtract numbers are both 3, and for all travel times 30 and above, the add/subtract numbers are both 5.

The user can use the “add more rows” link if more than 3 different sets of add/subtract numbers are desired. The Examples column shows an example of the travel time range created for a travel time using the add/subtract numbers on each row. To edit the settings, the user makes entries in each field for the rows they wish to use, and blanks all fields on rows they wish to remove, and clicks Submit. The settings take effect immediately and will be applied the next time a travel time range field is populated in a travel time / toll rate message.

5.1.5.2 System-Wide Travel Time Message Schedule

The system-wide travel time message schedule is used for DMSs whose travel time message schedule is set to use the system default (no custom schedule). It can either specify that there are no restrictions on the display of travel time messages (24x7), or sets specific time periods when travel time messages may be displayed.

CHART - Windows Internet Explorer

Comm Log Source Other (no info) Text Add I/S C

[Toggle Menu](#) | [Recent Events](#) | [Back](#) | [Forward](#) | [Refresh](#) | [Center Rpt](#) | [Communications Log](#) | [Instant Messaging](#) | [Home Page](#) | [Paging](#) | [Map](#) | [Traffic Events](#) | [Help](#)

Edit Travel Time Message Display Schedule

(Does not apply to travel time messages that also contain toll rate)

☐ Enabled 24/7 ☒ Enabled during specific times

Travel Time Message Allowed From: 05 : 30 Until: 09 : 30

Travel Time Message Allowed From: 15 : 00 Until: 18 : 30

[Add Time Range](#)

[Top](#) | [Back](#) | [Forward](#) | [Refresh](#) | [Center Rpt](#) | [Communications Log](#) | [Instant Messaging](#) | [Home Page](#) | [Paging](#) | [Map](#) | [Traffic Events](#) | [Help](#) | [Save Window Position](#)

CHART R3B3 Prototype 11/12/2008 © 2002-2008 MDSHA. All rights reserved.

Local intranet 100%

Figure 5-44 System Wide Travel Time Message Schedule

When entering specific time periods, the user selects a start time and end time for each period when travel time messages may be displayed. The “Add Time Range” link can be used to add additional time periods.

5.1.5.3 Miscellaneous Travel Time Settings

Several miscellaneous travel time settings exist in the system profile (see below).

CHART - Windows Internet Explorer

Comm Log Source Other (no info) Text Add I/S O/S Search

[Toggle Menu](#) | [Recent Events](#) | [Back](#) | [Forward](#) | [Refresh](#) | [Center Rpt](#) | [Communications Log](#) | [Instant Messaging](#) | [Home Page](#) | [Paging](#) | [Map](#) | [Traffic Events](#) | [Help](#)

Miscellaneous Travel Time Settings

Travel Time Trend Threshold This setting specifies the % change for defining up / down / flat trends for travel times on routes. The newest N travel times for a route must differ from the oldest N travel times for the route by the specified percentage for the trend to be considered up or down. If the difference is negative (newer less than older by the specified percentage), then the trend will be down. If the difference is positive (newer greater than older by the specified percentage), then the trend will be up. N is defined in the Travel Time Trend Sample Size setting below.

1 %

Travel Time Trend Sample Size This setting specifies the number of travel times that are averaged together when calculating the travel time trend. This number will only be used if it is smaller than 1/2 the number of travel time history entries kept for a travel route.

Travel Time Expiration This setting specifies the amount of time (in minutes) that a travel time is considered to be valid. This setting should be set higher than the expected travel time update frequency.

[Top](#) | [Back](#) | [Forward](#) | [Refresh](#) | [Center Rpt](#) | [Communications Log](#) | [Instant Messaging](#) | [Home Page](#) | [Paging](#) | [Map](#) | [Traffic Events](#) | [Help](#) | [Save Window Position](#)

CHART R3B3 Prototype 11/12/2008 © 2002-2008 MDSHA. All rights reserved.

Local intranet 100%

Figure 5-45 Miscellaneous Travel Time Settings

The **Travel Time Trend Threshold** setting specifies the percent change required in the recent travel times for a travel route for the system to consider the trend up or down. If the change is less than this percent, the trend is considered flat.

The **Travel Time Trend Sample Size** setting specifies the number of travel times to average together when determining the trend. This number of latest recent travel times will be averaged and compared to the average of the earliest recent travel times for a travel route to determine the percent change. The Travel Time Trend Threshold will then be used to determine if the trend is flat or up/down.

The **Travel Time Expiration** setting specifies the amount of time a travel time is considered to be valid. A travel time that has not been updated within this timeframe will be considered expired.

5.1.6 Geographical Settings

Geographical settings include the management of geographical areas and coordinate sanity check limits. Geographical areas are named polygons that are used in external event import rules as

well as external device query rules. They may be used in a future release as the basis for areas of responsibility. The coordinate sanity check limits are settings that are used to make sure that user entered latitude and longitude values are reasonable. The geographical settings are available in the Geographical Settings section of the system profile page as shown below.

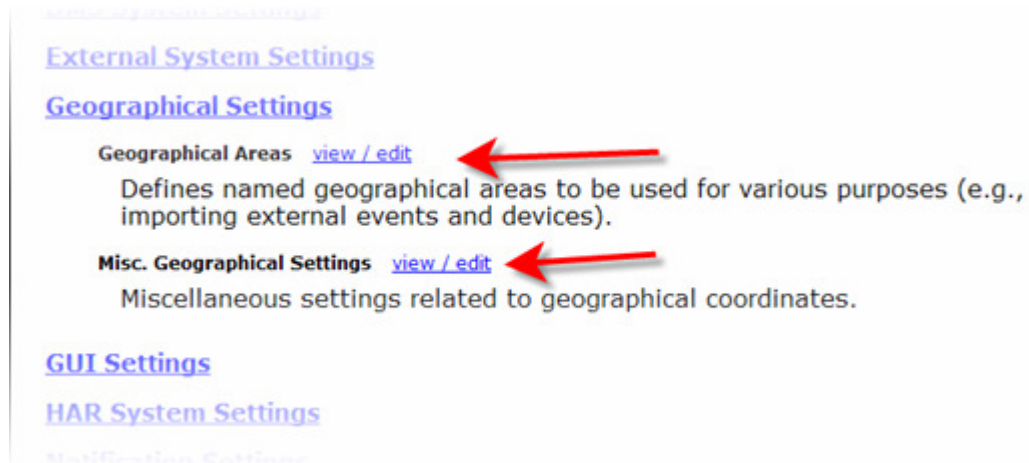


Figure 5-46 Geographical Settings

See the sections below for details on the Geographical Settings.

5.1.6.1 Geographical Areas

Geographical areas are used as filter criteria in external event import rules and in queries used to manage external devices. When accessed via the system profile page, the list of currently defined geographical areas is shown.

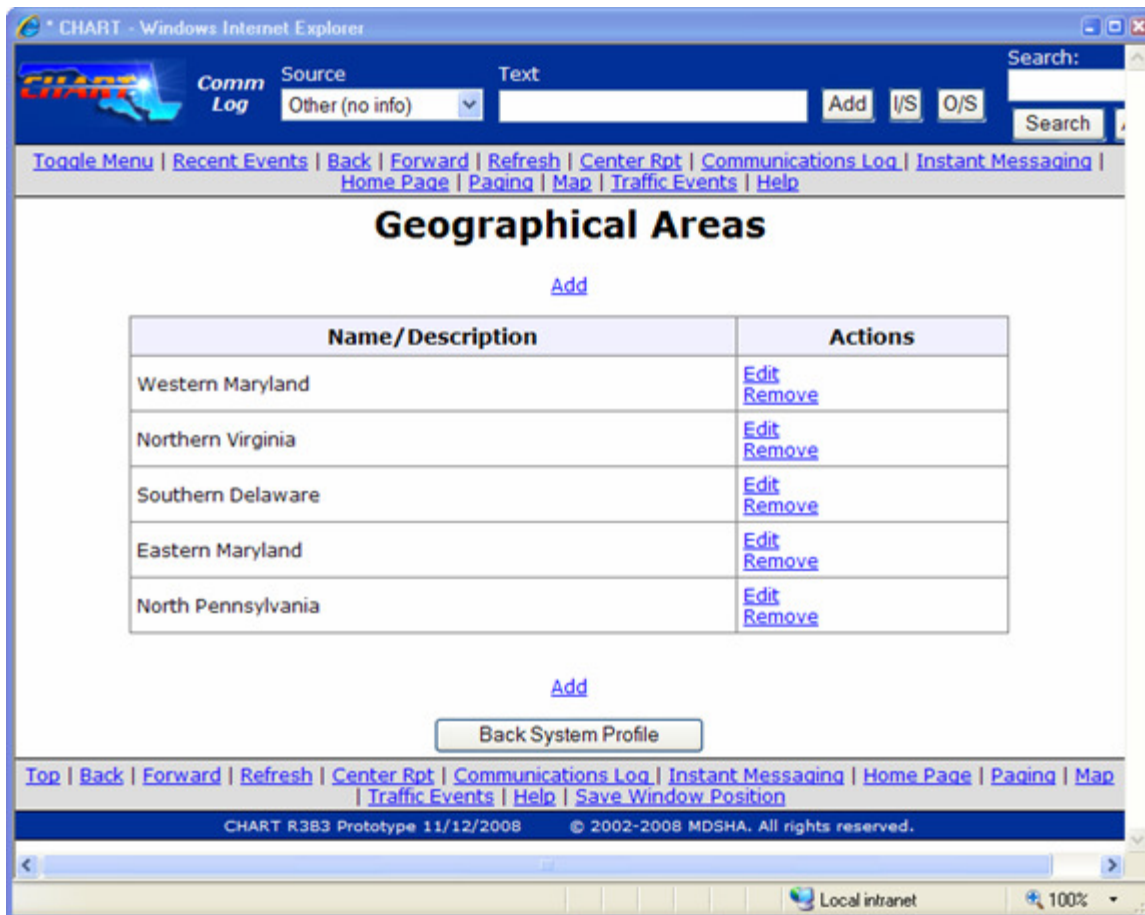


Figure 5-47 Geographical Areas – List

The list shows each geographical area's name and provides links for each area to allow it to be edited or removed. Links at the top and bottom of the page allow new areas to be added. The Back To System Profile button navigates back to the user profile page. The sections below provide more details about managing geographical areas.

5.1.6.1.1 Add Geographical Area

To add a geographical area, the user needs to specify a name for the area and the points that make up the polygon that defines the area. Rather than (or in addition to) adding points manually, points can be imported from a KML file, a file format supported by most popular mapping products.

CHART - Windows Internet Explorer

Comm Log Source: Other (no info) Text:

[Toggle Menu](#) | [Recent Events](#) | [Back](#) | [Forward](#) | [Refresh](#) | [Center Rpt](#) | [Communications Log](#) | [Instant Messaging](#) | [Home Page](#) | [Paging](#) | [Map](#) | [Traffic Events](#) | [Help](#)

Add Geographical Area

Use this form to Add/Edit a Geographical Area

Name/Description:

Select KML Area Import File (Optional):

Location Points:

Latitude(Decimal Degrees) Ex. 38.664872	Longitude(Decimal Degrees) Ex. -77.134821	
<input type="text"/>	<input type="text"/>	Insert Point
<input type="text"/>	<input type="text"/>	Insert Point
<input type="text"/>	<input type="text"/>	Insert Point
Add Another Point		

[Top](#) | [Back](#) | [Forward](#) | [Refresh](#) | [Center Rpt](#) | [Communications Log](#) | [Instant Messaging](#) | [Home Page](#) | [Paging](#) | [Map](#) | [Traffic Events](#) | [Help](#) | [Save Window Position](#)

http://localhost:8080/chartite/app?action=getExtS Local intranet 100%

Figure 5-48 Add Geographical Area

To import points from a KML file, the user must first generate the KML file with the mapping product of their choice and store it on the computer they are using to access CHART. The user can then use the Browse button to navigate to the file on their computer and select it. After the file is selected, the user must click the “Import Points From File” button. The points in the KML file will be used to populate the latitude and longitude fields in the Location Points table, with rows being added to the table as needed. After points are imported, the user may still choose to make manual point entries, or they can use the Add Area button to add the area as defined via the imported points.

When manually entering points, the user can use the Insert Point link to insert fields for a new point above the row where they click the Insert Point link. They can use the Add Another Point link to add fields for another point below the current list of points.

After the list of points is complete, the user can click the Add Area button to add the area to the system. The Geographical Areas list will be shown and the new area will be included in the list.

5.1.6.1.2 Edit Geographical Area

The user may edit an existing geographical area using the “Edit” link for the area in the geographical area list. The same form used to add an area (see above) will be shown; however it will be populated with the name of the area and the points currently included in the area. The user can then make any edits necessary and submit the form to save their changes.

5.1.6.1.3 Remove Geographical Area

The user may remove a geographical area using the “Remove” link for the area in the geographical area list. A warning message will appear to help prevent accidental removal, and if the user confirms their intention to remove the area the geographical area list will be refreshed and the removed area will no longer appear in the list.

5.1.6.2 Miscellaneous Geographical Settings

The miscellaneous geographical settings include minimum and maximum latitude and longitude values used to check user lat/long entries to make sure they are reasonable. These could be set to the full allowed range of latitude and longitude for the world (-180, 180, -90, 90), or could be set to a geographical region closer to the area of operation encompassed by the CHART system (such as the mid-Atlantic states).

CHART - Windows Internet Explorer

Comm Log Source Text
Other (no info) Add

[Toggle Menu](#) | [Recent Events](#) | [Back](#) | [Forward](#) | [Refresh](#) | [Center Rpt](#) | [Communications Log](#) | [Instant Messaging](#) | [Home Page](#) | [Paging](#) | [Map](#) | [Traffic Events](#) | [Help](#)

Geographic Settings

Geo Coordinate Sanity Check Limits (decimal degrees)

These limits are used to ensure that coordinates entered by users are reasonable.

Min Longitude (western boundary)

Max Longitude (eastern boundary)

Min Latitude (southern boundary)

Max Latitude (northern boundary)

Done Local intranet 100%

Figure 5-49 Miscellaneous Geographical Settings

5.1.7 External Events

External events were added to the CHART system in R3B2. This feature is enhanced in R3B3 to allow rules to be defined to control which events are imported into the system. These rules are also used to trigger actions related to external events that are imported, such as firing an alert and/or notification and marking an external event as interesting. The list of existing external event import rules can be viewed and rules can be added, edited, and removed. See the sections below for details.

5.1.7.1 View External Event Inclusion Rules

The external event inclusion rules page is accessed via the view / edit link for External Event Rules in the External System Settings section of the system profile (see below).

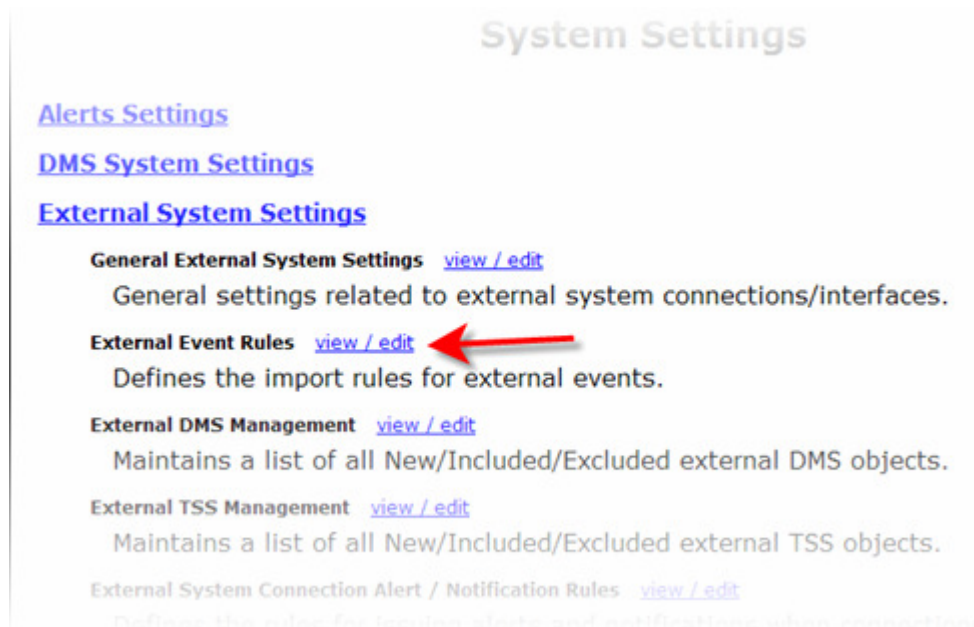
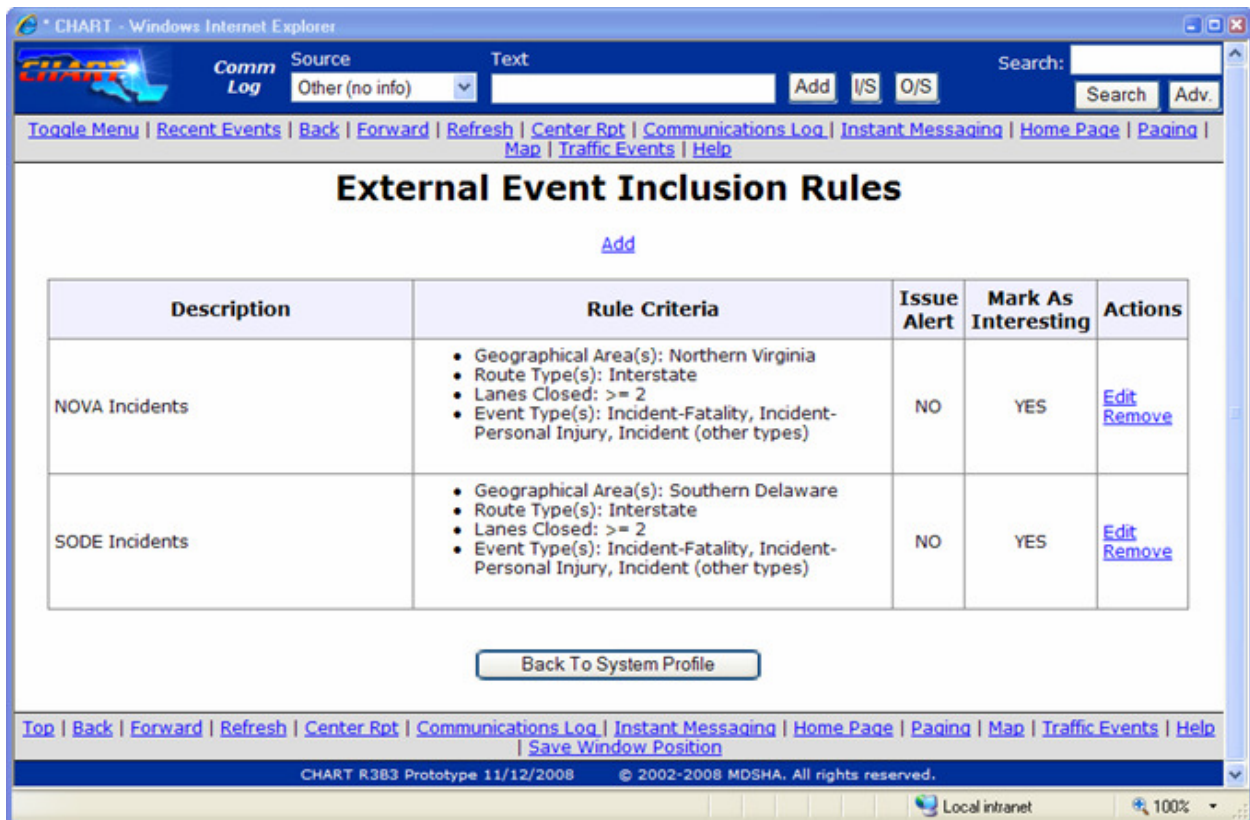


Figure 5-50 External Event Rules Link

After clicking the view/edit link, the External Event Inclusion Rules page is shown and all existing rules are listed. Any external event that meets any of the rules will be imported into the CHART system. If a rule is matched, other rules are still inspected to see if the import of the event should trigger an action. The name of each rule is shown along with the rule criteria and the actions associated with each rule (see below).



Links are included in the list to allow each rule to be edited or removed. A link at the top of the page allows a new rule to be added. The Back To System Profile button can be used to return to the system profile page.

5.1.7.2 Add External Event Inclusion Rule

When the Add link is clicked on the External Event Inclusion Rules page, the Add External Event Inclusion Rule page is shown. Each rule must be given a name and one or more criteria. Rules can optionally specify one or more actions to be taken when an event is imported that matches the rule.

CHART - Windows Internet Explorer

Comm Log Source Text Search: [] [Add] [I/S] [O/S] [Search] [Adv.]

[Toggle Menu](#) | [Recent Events](#) | [Back](#) | [Forward](#) | [Refresh](#) | [Center Rpt](#) | [Communications Log](#) | [Instant Messaging](#) | [Home Page](#) | [Paging](#) | [Map](#) | [Traffic Events](#) | [Help](#)

Add External Event Inclusion Rule

Use this form to Add/Edit an External Event Inclusion Rule

Name/Description:	<input style="width: 90%;" type="text"/>		
Rule Criteria			
Geographical Area(s):	<input type="text" value="--Select to Add--"/> [v]		
State(s):	<input type="text" value="--Select to Add--"/> [v]		
Route Type(s):	<input type="text" value="--Select to Add--"/> [v]		
Closed Lanes (>=):	<input type="text" value="--Any--"/> [v]		
Event Types:	<div style="display: flex; justify-content: space-between;"> <div style="width: 48%;"> <input type="checkbox"/> Incident - Fatality <input type="checkbox"/> Incident - Personal Injury <input type="checkbox"/> Incident - Other <input type="checkbox"/> Action <input type="checkbox"/> Safety Message </div> <div style="width: 48%;"> <input type="checkbox"/> Congestion <input type="checkbox"/> Weather <input type="checkbox"/> Disabled Vehicle <input type="checkbox"/> Special <input type="checkbox"/> Planned Closure </div> </div>		
Search Text: (Match one or more of: Name, Description, Route Number, County)	<input style="width: 80%;" type="text"/> More		
Rule Actions			
Issue Alert:	<input type="checkbox"/>		
Send Notification:	<input type="checkbox"/>		
Mark as Interesting:	<input type="checkbox"/>		
<input type="button" value="Add Rule"/> <input type="button" value="Cancel"/>			

Done Local intranet 100%

Figure 5-51 Add External Event Inclusion Rule

An external event must meet all criteria specified to match the rule. An event matching the rule will be imported into the CHART system and will cause any actions specified in the rule to be performed. The following criteria can be specified:

- **Geographical Area(s)** – One or more geographical areas (see section 5.1.6.1 above). An external event whose lat/long falls within any of the specified geographical areas will satisfy this criterion. A value of “Any” can be selected and will match any location specified within an external event (including those with no location specified). A special value of “Empty/Unspecified” may be selected and will match external events without a location specified.

- **State(s)** – One or more state codes. An external event that has a state code equal to any of the state codes specified will match this criterion. A value of “Any” can be selected and will match external events with any state specified (including those with no state specified). A special value of “Empty/Unspecified” can be selected and will match external events with no state specified.
- **Route Type(s)** – One or more route types on which an external event may be located, for example Interstate, US Route, State, etc. An external event that has a location with a route type equal to any of the route types specified will match this criterion. A value of “Any” can be selected and will match any route type included in the external event (including those with no route type specified). A value of “Empty/Unspecified” can be selected to match external events that have no route type specified.
- **Closed Lanes** – The number of lanes closed indicated by the event. An external event that has at least the specified number of lanes closed will match this criterion. A value of “Any” can be specified to match external events with any number of lanes closed (including those without the number of lanes closed specified). A value of “Empty/Unspecified” can be selected to match external events that don’t have the number of closed lanes specified.
- **Event Types** – The event types that will match this rule. If an external traffic event has an event type that will become any of the selected CHART event types after import, the external event will be included. If no event types are selected, external events with any event type will match this criterion.
- **Search Text** – One or more text strings that can be found in the event’s name, description, route number, or county. If any of the strings entered are found in any of the fields listed, the external event will match this criterion. If no text strings are entered, a text search will not be included when checking to see if an external event matches the rule.

The following actions can be specified in the rule:

- **Send Alert** – If this action is specified in the rule, any external event matching the rule will cause an alert to be sent to a specified operations center. When the action is enabled, a list box will appear to allow the user to choose the operations center.
- **Send Notification** – If this action is specified in the rule, any external event matching the rule will cause a notification to be sent to a specified notification group. When the action is enabled a list box will appear to allow the user to select the notification group.
- **Mark As Interesting** – If this action is specified in the rule, any external event matching the rule will have its “interesting” flag set to true when the event is imported. This will cause the event to appear on the home page (in the external events tab) of all users.

5.1.7.3 Edit External Event Inclusion Rule

An existing external event inclusion rule can be edited by clicking the “Edit” link for the rule in the rule list. A form will appear that is nearly identical to the Add External Event Inclusion Rule form shown above and will be pre-populated with the rule’s current data. The user can make any

desired changes and submit the form. Changes to a rule take effect immediately and will be used the next time external event data is received (or retrieved) from an external system.

5.1.7.4 Remove External Event Inclusion Rule

An external event inclusion rule can be removed from the system by clicking the “Remove” link for the rule in the rule list. A warning will appear and the rule will be removed if the user confirms their intent to remove the rule. The rule list will refresh and the removed rule will no longer appear.

5.1.8 External Devices

R3B3 allows external devices to be imported into the CHART system. Unlike traffic events, which are imported automatically based on a set of rules, external devices must be explicitly included in the CHART system by an administrator. External DMS (signs) and TSS (detectors) are supported in R3B3. Lists of candidate external devices (DMS or TSS) are managed by the administrator to mark devices for inclusion in CHART. A search feature allows the administrator to manage small sets of devices at a time, based on geographic location and other criteria. In addition to allowing devices to be marked as included, the system also allows devices to be marked as excluded. A device marked as excluded will be removed from CHART if it was previously included. A device marked neither included nor excluded remains a candidate for inclusion and is not included in the system. The included flag, excluded flag, or lack of either flag can be used as search criteria to further assist the administrator in managing the lists of external device candidates. Once an external device is included in CHART it can be viewed on the appropriate device list page along with CHART devices of the same type. Filters on the list pages allow external devices to be shown or hidden, and also allow the external devices to be filtered by agency. See the sections below for more details.

5.1.8.1 Managing External Devices

External device management screens are accessed from the system profile in the External System Settings section (see below). The administrator can click the view/edit link for the type of external devices they wish to manage (DMS or TSS).

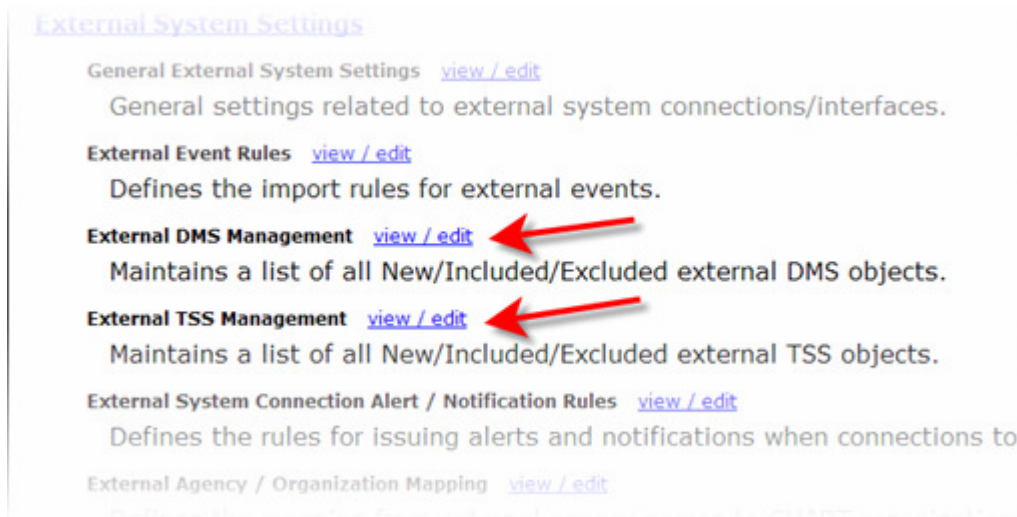


Figure 5-52 External Device Management Links

Aside from the device type being managed, management of external TSS and DMS devices is identical. Management of DMSs is shown in the examples below.

After clicking the device management view/edit link for the device type to be managed, a query screen appears (shown below) to allow the administrator to specify the group of devices they wish to manage.

CHART - Windows Internet Explorer

Comm Log Source Text Add I/S C

[Toggle Menu](#) | [Recent Events](#) | [Back](#) | [Forward](#) | [Refresh](#) | [Center Rpt](#) | [Communications Log](#) | [Instant Messaging](#) | [Home Page](#) | [Paging](#) | [Map](#) | [Traffic Events](#) | [Help](#)

Manage External DMSs

Enter Search Criteria

Agency:	--Select to Add--
Geographical Area(s):	--Select to Add--
Search Text: (device name/description, location description, county, route)	
Chart Applicability:	<input checked="" type="checkbox"/> New (not previously included/excluded) <input type="checkbox"/> Included (previously marked as included) <input type="checkbox"/> Excluded (previously marked as excluded)

Show DMS's
Cancel

[Top](#) | [Back](#) | [Forward](#) | [Refresh](#) | [Center Rpt](#) | [Communications Log](#) | [Instant Messaging](#) | [Home Page](#) | [Paging](#) | [Map](#) | [Traffic Events](#) | [Help](#) | [Save Window Position](#)

CHART R3B3 Prototype 11/12/2008 © 2002-2008 MDSHA. All rights reserved.

Local intranet 100%

Figure 5-53 Manage External DMSs - Query Page

The following search criteria may be entered.

- **Agency** – One or more agencies. Devices that are owned by any of the selected agencies will match this search criterion. A value of “Any” may be selected to match devices from any agency (including those without an agency specified). A value of “Empty/Unspecified” can be used to match devices without an agency specified.
- **Geographical Area** – One or more geographical areas. Devices that are located within any of the specified areas will match this search criterion. A value of “Any” may be selected to match devices in any location. A value of “Empty/Unspecified” can be selected to match devices that don’t have location data specified.
- **Search Text** – A text string to search for in the device name/description, location description, county, or route. If specified, only devices with the given text in at least one of these fields will match this criterion. If no text is entered, search text will not be included in the search.
- **CHART Applicability** – Flags used to select devices based on their previously set CHART applicability (included, excluded, or neither). At least one flag must be selected. All flags should be selected if the desire is to see all devices that match the other search

criteria. Only devices whose included/excluded flag settings match the selected applicability settings will match this criterion.

To view all external devices, the Agency and Geographical Area criteria should be set to Any, the search text should be left blank, and all CHART applicability flags should be selected. If this is done, the list of devices could be very large and therefore it is usually better to enter other search criteria.

After the criteria are entered, the administrator can click the “Show DMS’s” button to see the devices that match the specified criteria.

Manage External DMSs

☒ New ☒ Included ☒ Excluded

Description Δ / Location	Agency --ANY--	Included Include All	Excluded Exclude All
11417: I-95 Exit 27 SB I-95S @ Exit 27	MDOT	<input type="checkbox"/>	<input type="checkbox"/>
215A2: Advisory I-95 SB Past Exit 23 I-95S @ Exit 23	VDOT	<input type="checkbox"/>	<input type="checkbox"/>
215A3: Advisory I-95 SB Past Exit 63 I-95S @ Exit 63	DDOT	<input checked="" type="checkbox"/>	<input type="checkbox"/>
310C5: Advisory I-695 SB Past Exit 1 I-695S @ Exit 1	MDOT	<input type="checkbox"/>	<input checked="" type="checkbox"/>
31566: Advisory I-295 SB Past Exit 13 I-295S @ Exit 13	VDOT	<input type="checkbox"/>	<input type="checkbox"/>
		Include All	Exclude All

Save Cancel

Figure 5-54 Manage External DMSs - Search Results

The devices are shown along with their current included/excluded setting. The list can be sorted by Description, Location, Agency, Included Flag, or Excluded Flag, and can be filtered by Agency. The list can also be filtered using check boxes at the top to show or hide devices based on their current settings for the CHART applicability flags. The user can place check marks in the included or excluded box for each device, or both boxes can be left unchecked. The Include All and Exclude All buttons can be used to select all boxes in a column rather than having to select each individual box. When finished making their selections, the user must click the Save button to save their settings and cause the system to import newly included devices and remove

previously included devices that are no longer marked as included. The devices that are imported can be viewed in the appropriate device list.

5.1.8.2 Viewing External DMSs

External DMSs that have been marked as included and imported into the system can be viewed on the DMS List page along with CHART (internal) DMSs. The DMS List is accessed via the Message Signs link in the navigation area of the home page in the Device Management section.

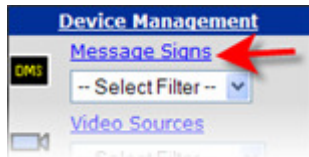


Figure 5-55 Message Signs Link on Home Page

After clicking the Message Signs link, the list of DMSs is shown. Initially, only CHART DMSs will appear. To view the external DMSs, the user must check the External DMSs box.

The screenshot shows a web browser window titled "Dynamic Message Signs - CHART - Windows Internet Explorer". The page displays a list of "Highway Message Signs (12)". At the top, there are filters for "Show: CHART DMSs" (checked) and "External DMSs" (checked). Below this, there are checkboxes for "External DMSs" from "DDOT", "DelDOT", "PennDOT", and "VDOT", all of which are checked. The table below lists the message signs with columns for Description, Message, Status, Route, Direction, County, and Used By. CHART DMSs have a light blue background, while external DMSs have a light green background.

Description / Location	Message	Status	Route	Direction	County	Used By
CHART DMS 4403 I-695 O/L (South), at Ex. 10 Alt US 1	[Redacted]	Online	I-999 South		County XYZ	
CHART DMS 00DOT DMS 0 DDOT Location 0	Sample Page 1 Sample Page 2	Online	I-999 South		County XYZ	
External DMS 7701 I-95 North, prior Ex 38 Md 32	[Redacted]	Offline	I-999 North		County XYZ	
External DMS DelDOT DMS 1 DelDOT Location 1	[Redacted]	Offline	I-999 North		County XYZ	
External DMS 7702 I-95 N, prior to Exit 43 Md 100	[Redacted]	Offline	I-999 North		County XYZ	
External DMS PennDOT DMS 2 PennDOT Location 2	[Redacted]	Offline	I-999 North		County XYZ	
External DMS 850 Washington Blvd (US Rt 1) NB @ Main Street	[Redacted]	Offline	I-999 North		County XYZ	
External DMS VDOT DMS 3 VDOT Location 3	[Redacted]	Offline	I-999 North		County XYZ	

Figure 5-56 DMS List with External DMSs

The external DMSs are shown with a different background color to allow them to be distinguished from CHART DMSs. It is also possible to hide CHART DMSs to show only external DMSs by unchecking the CHART DMSs box. When external DMSs are shown, checkboxes for each agency for which one or more DMS is included in the CHART system appear. The user can check a box to show devices from an agency or uncheck a box to hide devices from an agency.

5.1.8.3 Viewing External TSSs

External TSSs that have been marked as included and imported into the system can be viewed on the TSS List page along with CHART (internal) TSSs. The TSS List is accessed via the Detectors link in the navigation area of the home page in the Device Management section.

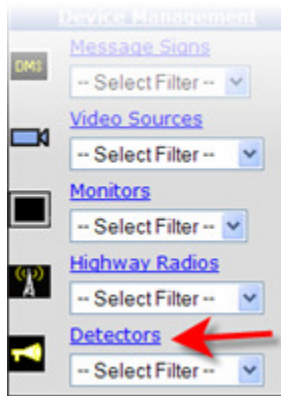


Figure 5-57 Detectors Link on Home Page

After clicking the Detectors link, the list of TSSs is shown. Initially, only CHART TSSs will appear. To view the external TSSs, the user must check the External Detectors box.

The screenshot shows the 'Detectors (28)' page. At the top, there are filters for 'Source' (Other (no info)), 'Text', and a search bar. Below the filters, there are checkboxes for 'Show: CHART Detectors' (checked) and 'External Detectors' (checked). Under 'External Detectors', there are checkboxes for 'DDOT', 'DelDOT', 'PennDOT', and 'VDOT' (all checked). The table below lists the TSSs with columns: Description / Location, Speed (Low Avg), Route, Direction, County/State, Status, and Last Update.

Description / Location	Speed (Low Avg)	Route	Direction	County/State	Status	Last Update
500 test location	20 MPH	I-95	North	Baltimore County, MD	Online	15:20
DDOT Detector 0 DDOT Location 0	20 MPH	I-95	North	County, State	Online	15:20
500 test	22 MPH	I-95	North	Baltimore County, MD	Maintenance	15:18
DelDOT Detector 1 DelDOT Location 1	22 MPH	I-95	North	County, State	Maintenance	15:18
700 test	41 MPH	I-95	North	Baltimore County, MD	Maintenance	15:19
PennDOT Detector 2 PennDOT Location 2	41 MPH	I-95	North	County, State	Maintenance	15:19
Copy of 500 test location	No Data	I-95	North	Baltimore County, MD	Offline	12/01/08 09:09

Figure 5-58 TSS List with External TSSs

The external TSSs are shown with a different background color to allow them to be distinguished from CHART TSSs. It is also possible to hide CHART TSSs by unchecking the CHART Detectors box. When external TSSs are shown, checkboxes for each agency for which one or more TSS is included in the CHART system appear. The user can check a box to show devices from an agency or uncheck a box to hide devices from an agency.

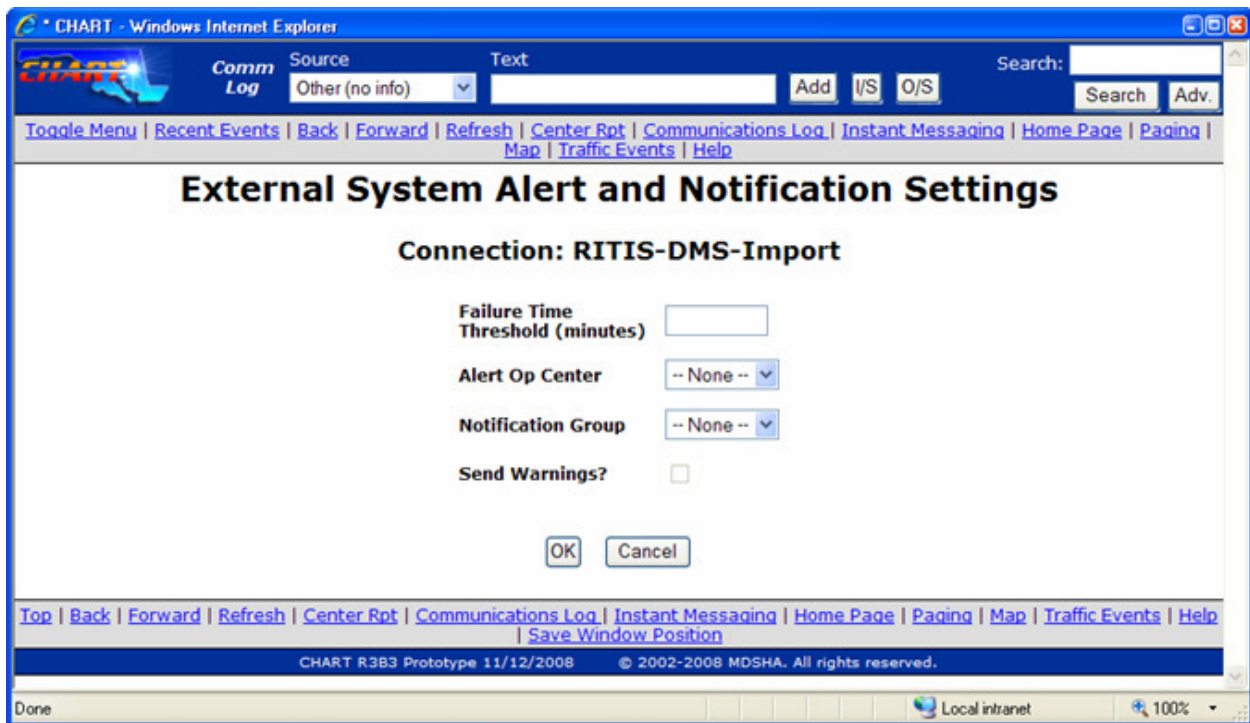


Figure 5-60 External System Alert and Notification Settings - Edit

5.1.9.2 External Agency / Organization Mapping

The External Agency to Organization Mapping configuration defines the CHART organization to be assigned to external events and devices when they are imported into CHART. The organization to use is based on the identification of the external system and the system's agency designation present in the external object's data.

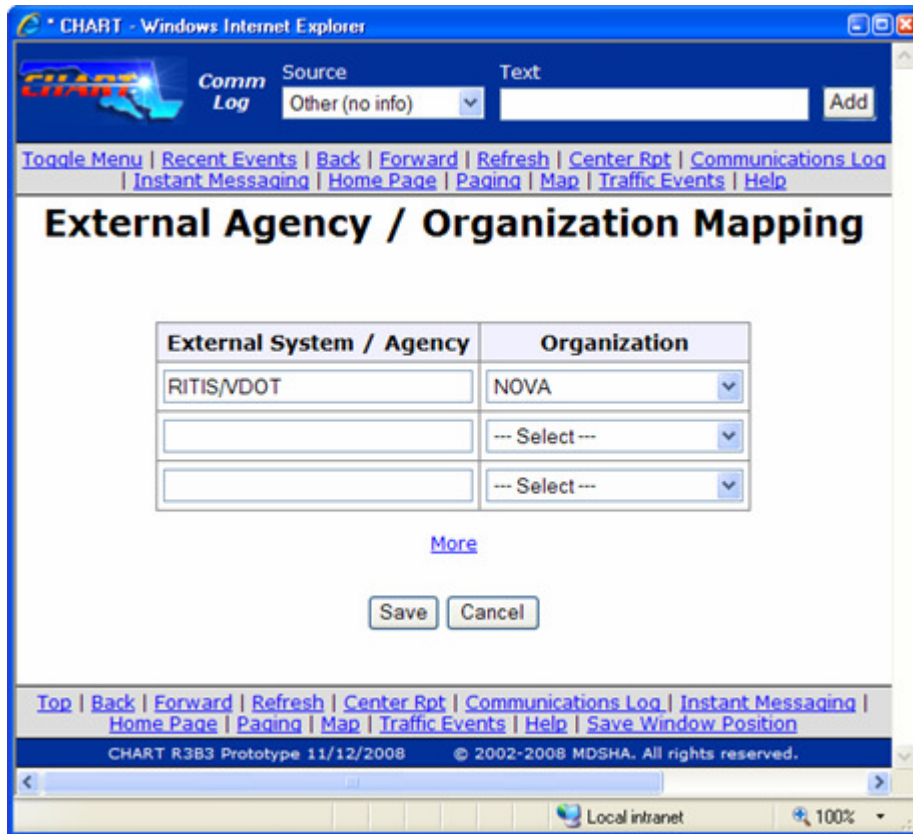


Figure 5-61 External Agency / Organization Mapping

5.1.9.3 External Client Management

CHART R3B3 allows external systems to connect to CHART to access its data. CHART R3B3 also allows one system (Vector) to connect to CHART to provide data (toll rates). The External Client Management feature is used to manage the credentials of these systems to ensure that only approved external systems can connect to CHART. The current list of approved external clients can be viewed, settings for existing clients can be edited, and clients can be removed. See the sections below for details.

5.1.9.3.1 View External Clients

The list of currently defined external clients is available using the view/edit link for External Client Management in the External System Settings section of the system profile (see below).



Figure 5-62 External Client Management - view/edit Link

After clicking the link the list of external clients is shown. For each client, the following information is shown:

- **Client ID** –The ID the client passes to CHART when it accesses the system. CHART uses the ID to find the client’s public key to verify an electronic signature required for all data transmissions to CHART. The ID is also used apply the proper user rights to the client with respect to providing access to data.
- **Client Name** – The name of the client.
- **Description** – The description of the client.
- **Access** – The type of access the client requires. A supplier provides data to CHART and is not subject to user rights. Consumers read data from CHART and are required to have one or more CHART Roles assigned that determine the user rights used to determine which data the external client can and cannot access.
- **Contact Info** – Contact information for the person responsible for the external system’s connection to CHART.

External System Client Management

[Add Client](#)

Client ID	Client Name	Description	Access	Contact Info	Actions
vecsys1	Vector	Provides toll rate information To Chart	<input checked="" type="checkbox"/> Supplier <input type="checkbox"/> Consumer	Name: Tom FenderBender Phone: 481-234-1234 Email: Tom@traffic.com	Edit Remove
ritisys1	RITIS	Retrieves detector and DMS data from CHART	<input type="checkbox"/> Supplier <input checked="" type="checkbox"/> Consumer Role(s): Monitor Traffic Monitor DMS Monitor TSS	Name: Rick Rubberneck Phone: 411-424-3221 Email: rick@mapquest.com	Edit Remove

[Add Client](#)

Top | Back | Forward | Refresh | Center Rot | Communications Log | Instant Messaging | Home Page | Paging | Map | Traffic Events | Help | Save Window Position

CHART R3B3 Prototype 11/12/2008 © 2002-2008 MDSHA. All rights reserved.

Figure 5-63 External Client List

The Actions column provides links to allow an existing client’s settings to be edited and to allow an existing client to be removed (thereby revoking their access to the CHART system). The Add Client links at the top and bottom of the page allow an external client to be added. These actions are described further in the sections that follow.

5.1.9.3.2 Add External System Client

An external client must be added to the external client list before the client can gain access to the CHART system. The Add External System Client form is accessed using the Add Client link on the External System Client Management page.

The screenshot shows a web browser window titled "CHART - Windows Internet Explorer". The address bar shows "Local intranet". The page has a blue header with the CHART logo and navigation links: "Comm Log", "Source" (dropdown menu set to "Other (no info)"), "Text" (input field), "Add", "I/S", and "O/S". Below the header is a navigation bar with links: "Toggle Menu", "Recent Events", "Back", "Forward", "Refresh", "Center Rot", "Communications Log", "Instant Messaging", "Home Page", "Paging", "Map", "Traffic Events", and "Help". The main content area is titled "Add External System Client". It contains a "Client Information" section with the following fields: "Client ID:" (input field), "Client Name:" (input field), "Description:" (input field), and "Contact Info:" (a table with rows for "Name:", "Email:", "Phone:", and "Fax:", each with an input field). Below this is a "Public Key:" section with a text area containing a Sun RSA public key, 1024 bits, modulus: 1365261606132521818815031521537172578668637322505 1444975349568384477744431524191472726155830565319 0873498443228025907747186118671231210392655271529 0758519219617116307617435284074005403813852388711 1082631288272865524042079297259258996881579696909 8876870487325757132196877669297944021682942798356 429884214905147 public exponent: 65537. Below the public key is a "Download Private Key" button. At the bottom is an "Access:" section with two checkboxes: "Supplier" and "Consumer". At the very bottom are "Add Client" and "Cancel" buttons.

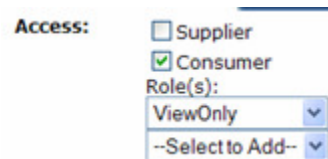
Figure 5-64 Add External System Client

The Add External System Client form allows the administrator to assign an ID to the external client which must be used by the client each time it accesses the CHART system. A name and description can also be assigned to further identify the client. Contact information including the contact's name, e-mail, phone, and fax may also be entered.

When the Add External System Client form is displayed, a public/private key pair is automatically generated. The public key is shown to the user and a button exists to allow the

private key to be downloaded. The private key is to be provided to the owner of the external system and must be used by the external system to electronically sign all data packets sent to the CHART system. This private key must be carefully guarded and securely provided to the external system owner. There is no need to store the private key within the CHART organization; a new public/private key pair can be generated should the external system owner lose the currently provided private key. CHART will utilize the public key shown to verify signatures created using the private key. Only the public key is stored in the CHART system.

The Access checkboxes allow the administrator to specify the type of access required by the external client. A supplier of data connects to CHART to provide data to CHART, such as toll rate data. A consumer connects to CHART to access CHART data. A client can be a supplier, consumer, or both. When the consumer checkbox is checked, a list appears to allow the administrator to assign a role to the external client.



Access:

☐ Supplier

☒ Consumer

Role(s):

ViewOnly

--Select to Add--

Figure 5-65 Set External Client Role(s)

After a role is chosen, another list will appear to allow another role to be added. An external client that is a consumer of CHART data can have 1 or more roles. A union of the user rights from all assigned roles will be used to determine the CHART data the external client is permitted to access.

5.1.9.3.3 Edit External Client

An external client can be edited using the Edit link for the client on the External System Client Management page. Clicking the Edit link causes the Edit External System Client page to appear, pre-populated with data for the external client. The Edit External System Client page differs slightly from the Add page with regard to the public/private keys. While the public key is shown as is the case on the Add form, the Download Private Key button is disabled, as CHART does not store the private key that is provided to the external system owner. If the external system owner requires a new private key (or the administrator forgot to download the private key generated while adding the external client), the admin may use the Generate New Keys button.

Figure 5-66 Edit External System Client

The Generate New Keys button, when pressed, will cause the public key text area to show a new public key, and the Download Private Key button will be enabled to allow the associated private key to be downloaded and provided to the external system owner. The Edit Client button must be clicked when finished making any changes to the external client information, including generating new keys.

5.1.9.3.4 Remove External Client

An external client can be removed using the Remove link for the client on the External System Client Management page. Removing a client immediately revokes the client's access to the

CHART system. A warning message is used to confirm an administrator's intent to remove an external client.

5.1.10 External System Connection Status

A page showing the status of each external system connection is available for users with appropriate rights. This status page is accessible using the External Connections link in the Administration section of the home page navigation links (shown below).

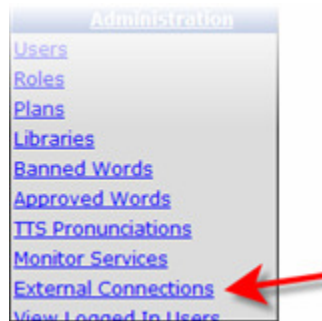


Figure 5-67 External System Connection Status – Link

After clicking the External Connections link, the External Connection Status page is shown. This page contains a row for each connection CHART makes to external systems, as well as potential connections that can be made from external systems to CHART. The potential incoming connections are based on external clients that have been configured to access the CHART system. The screen shot below depicts an example of the external connections that may appear on this status page.

A screenshot of a web browser window titled 'External System Status - CHART - Windows Internet Explorer'. The page has a blue header with the CHART logo and navigation links. Below the header is a table titled 'External System Connections'. The table has four columns: System, Status, Status Changed, and Status Confirmed. It lists nine connections, all with a status of 'OK'. At the bottom of the table is a 'Back' button. The browser's address bar shows the URL 'http://localhost:8080/chartite/app?action=viewEventList&sh'.

System	Status	Status Changed	Status Confirmed
CHART-Web-Export	OK	09:13	09:18
INRIX-Import	OK	09:13	09:18
RITIS-Detector-Export	OK	09:13	09:18
RITIS-Detector-Import	OK	09:13	09:18
RITIS-DMS-Export	OK	09:13	09:18
RITIS-DMS-Import	OK	09:13	09:18
RITIS-Event-Export	OK	09:13	09:18
RITIS-Event-Import	OK	09:13	09:18
Vector-Import	OK	09:13	09:18

Figure 5-68 External System Connection Status

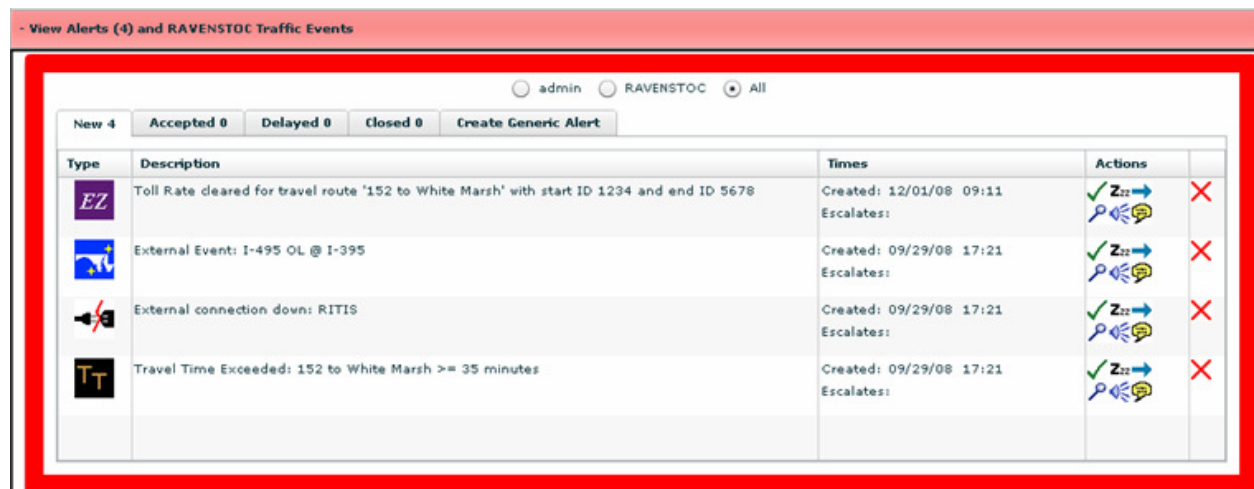
The current status of each connection is shown in addition to the time the status transitioned to the current state. The Status Confirmed time shows when the system last verified the current status.

5.1.11 Alerts

Four new alert types are added in R3B3: External Connection Alert, External Event Alert, Travel Time Alert, and Toll Rate Alert. All functionality regarding managing these new alert types is identical to the Alert functionality that exists in CHART R3B2 except as described in this section.

5.1.11.1 Alerts on the Home Page

Like other alerts, the alerts added in R3B3 appear on the home page of users in the operations center that receives the alert. Alerts for other centers can be viewed by users that have the proper rights and select the “All” alert filter on their home page.



























Type	Description	Times	Actions
	Toll Rate cleared for travel route '152 to White Marsh' with start ID 1234 and end ID 5678	Created: 12/01/08 09:11 Escalates:	   
	External Event: I-495 OL @ I-395	Created: 09/29/08 17:21 Escalates:	   
	External connection down: RITIS	Created: 09/29/08 17:21 Escalates:	   
	Travel Time Exceeded: 152 to White Marsh >= 35 minutes	Created: 09/29/08 17:21 Escalates:	   

Figure 5-69 R3B3 Alerts on the Home Page


Each of the alerts new to R3B3 appears in the image above. The icons used to represent each new alert type are as follows:

- This icon:  is used for external connection alerts. This type of alert is fired when the system detects an error related to an external connection. Warnings related to external connections may also generate alerts depending on the system configuration. See the section above on External System Connection Alert / Notification Rules for details.
- This icon:  is used for external event alerts. This type of alert is fired when an external event imported into CHART matches a rule that has the Send Alert action enabled. See section 5.1.7.2 above for details.

- This icon:  is used for toll rate alerts. The EZ is representative of EZPass, the electronic tolling system used in Maryland. This type of alert is fired when error conditions related to toll rates are detected, such as the toll rate for a travel route missing from the toll rate data feed.
- This icon:  is used for travel time alerts. This type of alert is fired when the travel time for a travel route exceeds an “alert travel time” for the route and travel time alerts are enabled for the route. See section 5.1.1.2 above for details.

All actions available on the home page for alert types that exist in R3B2 apply to these alert types added in R3B3.

5.1.11.2 Alert Details

The alert details page for an alert is accessible by clicking the  icon for the alert in the Actions column of the alert list. Each details page contains fields specific to the alert type in addition to elements common to all alert types. Only the fields specific to the new R3B3 alert types are discussed below.

5.1.11.2.1 External Connection Alert Details

The fields on the external connection alert details page that are specific to external connection alerts are shown below:

Current Status	FAILED
Current Status Change Time	14:25
Current Status Confirmed Time	14:37
Alerted Status	FAILED
Alerted Status Change Time	13:15
Alerted Status Confirmed Time	13:25

Figure 5-70 External Connection Alert Details - Type Specific Fields

These fields show the current status of the connection in addition to the status of the connection at the time the alert was issued. The status change time is the time the status of the connection transitioned to its current state. The status confirmed time is the time the system last detected the reported status.

5.1.11.2.2 External Event Alert Details

The fields on the external event alert details page that are specific to external event alerts are shown below:

External Event	I-495 OL @ I-395
Matching Rule	Major NoVA 495 Incidents
	<ul style="list-style-type: none"> • Area: NoVA • Route Type: Interstate • Event Type: Incident • Search Text: 495 • Lane Closures: >= 2

Figure 5-71 External Event Alert Details - Type Specific Fields

The External Event field shows the external event that triggered the alert and provides a link to the details page for the external event. The Matching Rule field shows the criteria in the rule that the event matched which caused the alert to be fired.

5.1.11.2.3 Toll Rate Alert Details

The field on the toll rate alert details page that is specific to toll rate alerts is shown below:

Route	152 to White Marsh
-------	------------------------------------

Figure 5-72 Toll Rate Alert Details - Type Specific Fields

The Route field shows the travel route for which a toll rate alert was fired and provides a link to the details page for the route.

5.1.11.2.4 Travel Time Alert Details

The fields on the travel time alert details page that are specific to travel time alerts are shown below:

Route	152 to White Marsh
Most Recent Travel Time	32 minutes @ 13:37
Alerted Travel Time	37 minutes @ 13:13
Travel Time Alert Limit	35 minutes

Figure 5-73 Travel Time Alert Details - Type Specific Fields

The Route field shows the travel route whose travel time exceeded the Alert Travel Time specified for the travel route. The Most Recent Travel Time field shows the current travel time for the travel route, and the Alerted Travel Time field shows the travel route's travel time at the time the alert was fired. The Travel Time Alert Limit field shows the Alert Travel Time that was configured for the travel route at the time the alert was fired.

5.1.11.3 Resolve Alert

The resolve action for an alert is performed by clicking the Resolve link on the alert's details page, or by clicking the ➡ icon for the alert in the home page alert list. Following are the resolve actions for each of the alert types added in R3B3:

- **External Connection Alert** – The External System Connections Status page is shown
- **External Event Alert** – The details page for the external event is shown

- **Toll Rate Alert** – The details page for the travel route for which the alert was generated is shown
- **Travel Time Alert** – The details page for the travel route whose travel time exceeded the Alert Travel Time is shown.

5.1.11.4 Alert Audio Cue Settings

The Alert Audio Cue settings page which exists in R3B2 is updated in R3B3 for the alert types added in R3B3. This page shows the current audio cue configured for each alert type (if any) and provides links to allow an audio cue to be added or for an existing audio cue to be edited or removed.

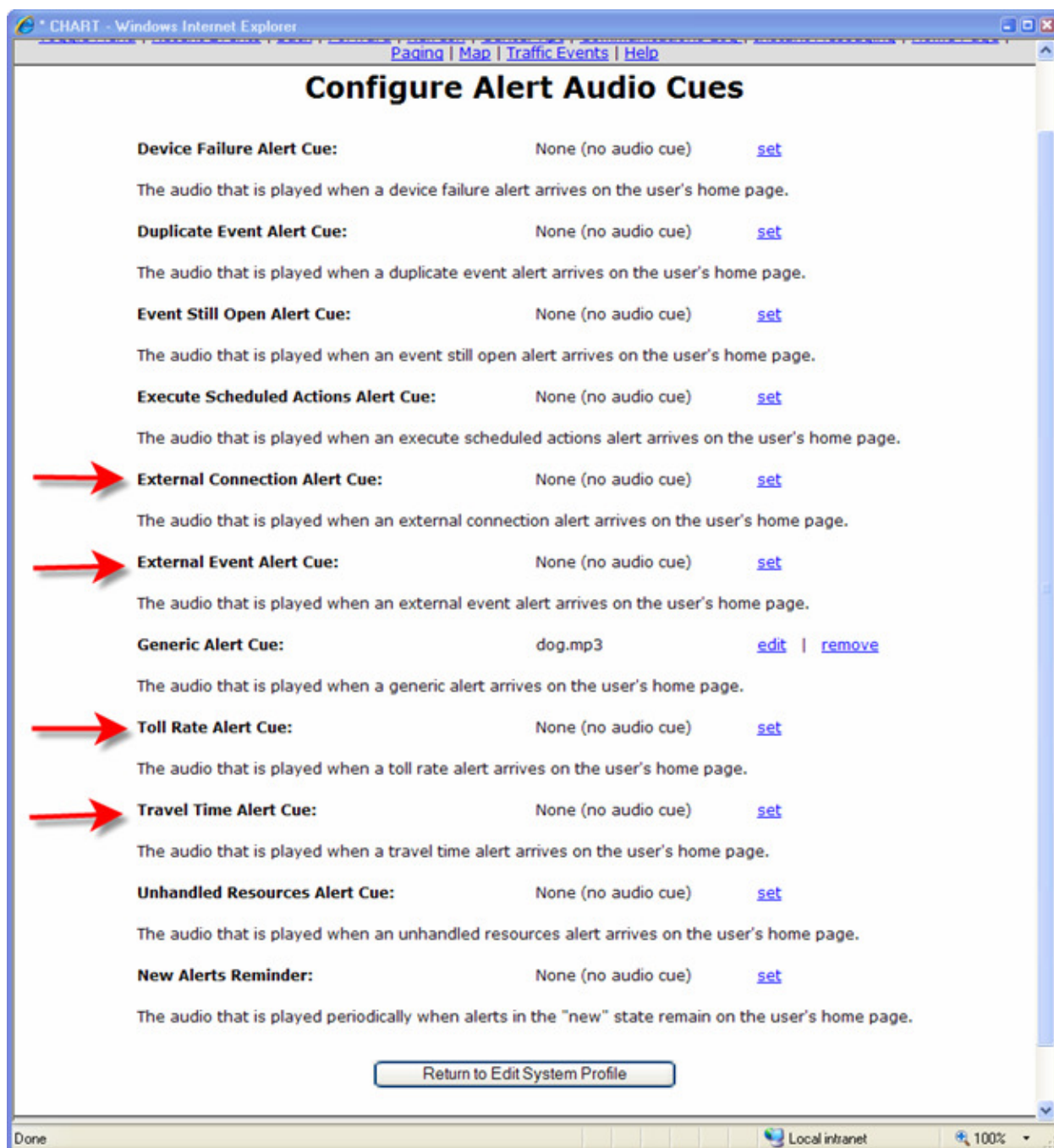


Figure 5-74 Configure Alert Audio Cues

5.1.11.5 Alert Timeouts and Policy Settings

The Alert Timeouts and Policy Settings page that exists in R3B2 is updated in R3B3 for the alert types added in R3B3. The page is very long due to the number of alert types – a portion of the page is shown below.

Travel Time Exceeded Alert

☒ Enable Travel Time Exceeded Alerts
☐ Disable Travel Time Exceeded Alert Auto Escalation

Timeout	Days	Hours	Minutes
Default Accept Timeout	0	0	30
Maximum Accept Timeout	28	0	0
Default Delay Timeout	0	0	30
Maximum Delay Timeout	28	0	0
Escalation Timeout	0	2	0

Toll Rate Alert Alert

☒ Enable Toll Rate Alert Alerts
☐ Disable Toll Rate Alert Alert Auto Escalation

Timeout	Days	Hours	Minutes
Default Accept Timeout	0	0	30
Maximum Accept Timeout	28	0	0
Default Delay Timeout	0	0	30
Maximum Delay Timeout	28	0	0
Escalation Timeout	0	2	0

External Connection Down Alert

☒ Enable External Connection Down Alerts
☐ Disable External Connection Down Alert Auto Escalation

Timeout	Days	Hours	Minutes
---------	------	-------	---------

Figure 5-75 Alert Timeout and Policy Settings

This page has the following settings for each alert type:

- **Enable/Disable the alert type** – Allows the alert type to be enabled or disabled, system-wide. When disabled, alerts of this type will not be generated. Alerts of this type that already exist in the system will not be affected.

- **Enable/Disable automatic escalation** – Allows automatic escalation for the alert type to be enabled and disabled. When enabled, the visibility for an alert in the “new” state will automatically increase to include all currently alerted center’s backup centers after the alert has been in the “new” state for the specified escalation timeout.
- **Default accept timeout** – The default amount of time an alert of this type will stay in the accepted state before automatically being moved back to the “new” state.
- **Maximum accept timeout** – The maximum accept timeout that is allowed for the alert type.
- **Default delay timeout** – The default amount of time an alert of this type will stay in the delayed state before automatically being moved back to the “new” state.
- **Maximum delay timeout** – The maximum delay timeout that is allowed for an alert type.
- **Escalation timeout** – The amount of time an alert of this type must stay in the “new” state before the alert is automatically escalated when automatic escalation is enabled.

5.1.12 Device Locations

Prior to R3B3, the only location data for field devices in the CHART system was a textual location description. In R3B3, detailed location fields are added to each field device. These location fields are a subset of the location fields that exist for traffic events. This subset includes the following location fields: State, County, Route Type, Route, Direction, Proximity, Intersecting Feature, Location Description, and Lat/Long. The intersecting feature includes roads and state milepost.

The field device types for which these detailed location fields are supported are DMS, Camera, HAR, Detector (TSS), and SHAZAM. (Monitors are not considered field devices). The location fields for a device can be set while adding the device to the system or as an edit operation from the device’s details page. The setting and viewing of device location fields is identical for each field device type – DMS is used as an example in the screen shots used in the detailed sections below.

A feature related to device locations that is also added in R3B3 is the ability to view devices that are close to a traffic event. This feature is made possible due to the addition of the lat/long fields for devices. Details on this feature in addition to viewing and setting device locations can be found in the sections that follow.

5.1.12.1 Setting Device Location Fields

Device location fields can be set when adding a device to the system or via an Edit link on the device’s details page. In either case, the form displayed will appear the same, however when performing an edit the location fields will be pre-populated with the current values for the device.

CHART - Windows Internet Explorer

CHART Main Window Help

Location Settings For: 4403

State: MD

County: Baltimore County

Route Type: I (Interstate)

Route: I-70

☐ Show Name

Direction: East

Proximity: AT

Feature Type: Road

Intersection:

☒ Show Name

Location Desc: I-70 EAST

Latitude: Format is decimal degrees. For example:
39.050731
Longitude: -76.930406

Submit Cancel

CHART R3B3 Prototype 11/12/2008 © 2002-2008 MDSHA. All rights reserved.

Figure 5-76 Location Settings Form

This form operates the same way as the location fields on the traffic event creation form. As the user selects fields, the select lists for the other fields update to include applicable entries. For example, when a county is selected, the list of routes is updated to include only routes of the selected type that exist in that county. This dynamic list population is only possible when the selected state is MD. Otherwise the select lists change to text entry fields and the data must be entered free-form.

The form creates a location description automatically as location fields are selected/entered; however the user can override the description text. All fields on the form are optional except for the location description. After making selections / entries, the user may hit the submit button to save their changes, or cancel to abandon their changes.

5.1.12.2 Viewing Device Location Fields in a Device List

The device lists for each field device type (excludes monitors) are updated in R3B3 to contain several location related columns (Route, Direction, Mile Post, and County). These columns include the ability to sort and filter. Additionally, each device list (including the list of monitors)

is enhanced to allow the user to choose the columns that are to be displayed or hidden. The addition of the location columns to the device lists greatly increased the width of these pages; the show/hide columns feature allows users to hide these new columns if they wish or hide other columns to control the width of the page and reduce or eliminate the need for horizontal scrolling.

The screenshot shows the CHART Dynamic Message Signs interface in a Windows Internet Explorer browser. The page title is "Highway Message Signs (6)". Below the title, there are links for "Add DMS" and "Set Columns". A "Show:" section has checkboxes for "CHART DMSs" (checked) and "External DMSs" (unchecked). Below this is a table with 7 columns: "Description / Location", "Message", "Status", "Route", "Direction", "County", and "Used By". Each column has a dropdown menu with "--Any--" selected. The table contains 6 rows of data, each representing a message sign. The first row is for sign 4403, located at I-695 O/L (South), at Ex. 10 Alt US 1, with status "Online", route "I-999 South", direction "County XYZ", and used by "County XYZ". The other rows are for signs 7701, 7702, 850, 853, and 855, all with status "Offline" or "Online", route "I-999 North", direction "County XYZ", and used by "County XYZ".

Description / Location	Message	Status	Route	Direction	County	Used By
I-695 O/L (South), at Ex. 10 Alt US 1		Online	I-999 South	County XYZ	County XYZ	
I-95 North, prior Ex 38 Md 32		Offline	I-999 North	County XYZ	County XYZ	
I-95 N, prior to Exit 43 Md 100		Offline	I-999 North	County XYZ	County XYZ	
Washington Blvd (US Rt 1) NB @ Main Street		Offline	I-999 North	County XYZ	County XYZ	
1116 Frankfurst Ave NB		Offline	I-999 North	County XYZ	County XYZ	
Shell Rd @ 1200 Chesapeake Ave		Online	I-999 North	County XYZ	County XYZ	

Figure 5-77 Device List with Location Columns

To set the column visibility, the user can click the Set Columns link to display the list of available columns with their current display status and can check/uncheck columns as desired to set their show/hide setting.

The screenshot shows the "Column Display" dialog box. It has a title bar "Column Display" and a list of columns with checkboxes. The columns are: "Description/Location" (checked), "Message" (checked), "Status" (checked), "Route" (checked), "Direction" (checked), "County" (checked), "Used By" (checked), "Port Managers" (unchecked), "Connection Site" (unchecked), "TT Schedule Overridden" (unchecked), "Trav Info Priority Overridden" (unchecked), "Owning Organization" (unchecked), and "Mile Post Marker" (unchecked). At the bottom are "Submit" and "Cancel" buttons.

Column	Visible
Description/Location	Yes
Message	Yes
Status	Yes
Route	Yes
Direction	Yes
County	Yes
Used By	Yes
Port Managers	No
Connection Site	No
TT Schedule Overridden	No
Trav Info Priority Overridden	No
Owning Organization	No
Mile Post Marker	No

Figure 5-78 Device List - Set Column Visibility

After submitting custom column display settings, the user can return to the default display by clicking the Show Default Columns link on the device list page.

Several other columns are added to the device lists such as Port Managers (if applicable), Network Connection Site, and Owning Organization. Additionally, columns have been added to the DMS list to show if the DMS has overridden the system-wide travel time display schedule, and whether or not the DMS is using standard arbitration queue priorities (buckets) for its travel time and toll rate messages.

5.1.12.3 Viewing Device Location Fields on a Details Page

The details page for each field device contains a section where location fields are displayed. The location description, which used to be shown in the Basic Settings section, has been moved to this new location section in R3B3.



Location: (Edit)	
Location Desc.:	
County:	Howard County
Route Type:	Interstate
Route:	I-95
Direction:	South
State Milepost:	76.5
Lat/Long:	39.197098° N, 76.766565° W

Figure 5-79 Location Fields on Device Details Page

An Edit link appears for users with appropriate rights to allow them to edit the location fields.

5.1.12.4 Viewing Devices Close to a Traffic Event

A new section is added to the traffic event details page in R3B3 to show devices that are located within a specified radius of the event. This new “close devices” section is initially collapsed to save vertical space on the page and can be expanded by the user to see a summary for each device type listed. The sub-sections for each device type can also be expanded to show actual devices. The close devices section of the traffic event details page is shown below with all sub-sections fully expanded.

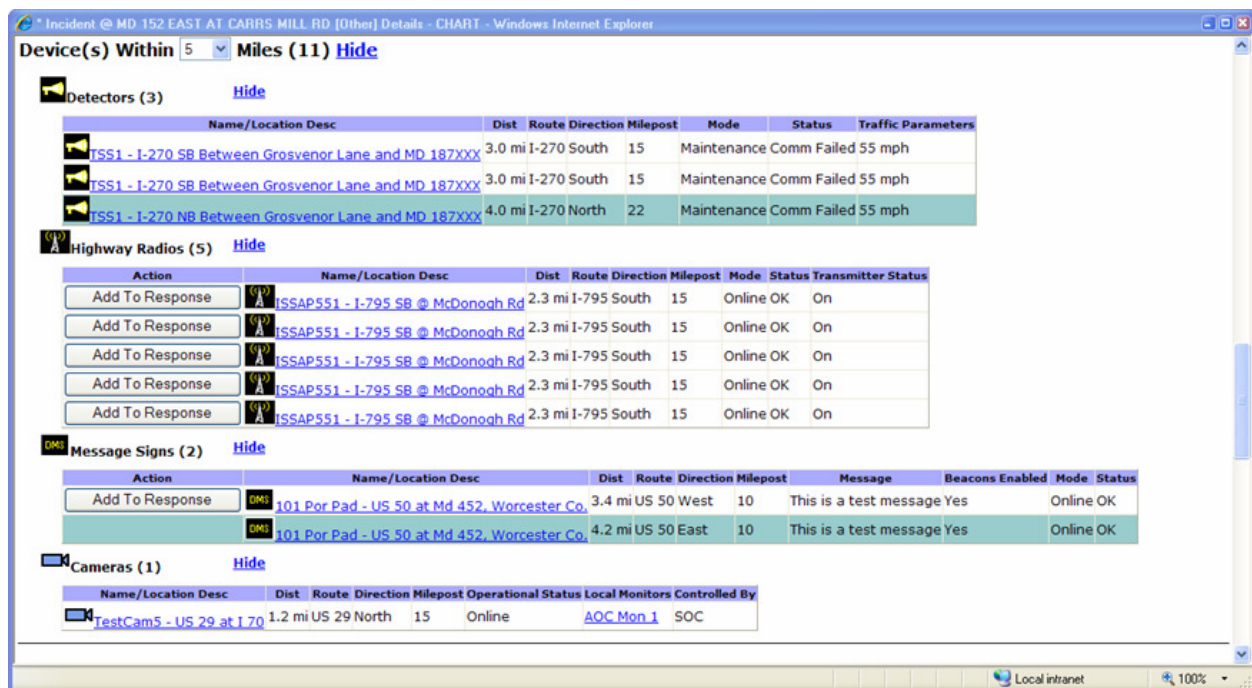


Figure 5-80 Devices Close to Traffic Event

The user may change the radius setting to expand or shrink the geographical circle around the event that is used to determine which devices are “close”. The system will save the user’s radius setting for the traffic event and will return to that setting if the page is refreshed or if the user navigates away from the page and returns.

External detectors and DMSs are included in the close device list and are shown with a shaded background color to differentiate them from internal CHART devices. While the user cannot control external devices, the user can see the current state of the device (such as the current speed of a detector or the current message of a DMS), and could contact the owning agency if device control is needed. Each DMS and HAR listed (with the exception of external DMSs) can be added to the response plan of the traffic event with an empty message using the Add To Response button that appears for these devices.

5.1.13 Miscellaneous Device Enhancements

R3B3 includes enhancements to allow TCP/IP communications for DMS and TSS devices, and to allow the default font to be set for NTCIP DMSs. The TSS details page is enhanced to show traffic parameters for individual zones within a zone group. See the sections below for details.

5.1.13.1 TCP/IP Device Communications

R3B3 includes support for TCP/IP communications for DMS and TSS devices. The form used to set the communication settings for these devices is updated to provide TCP/IP as a communication option for the device.

CHART - Windows Internet Explorer

CHART [Main Window](#) [Help](#)

Field Comm Settings For DMS: 4403

Drop Address (1-255)

Polling Enabled ☐

Use Comm Loss Timeout ☐

NOTE - Changing the comm loss timeout is not allowed because it requires communication to the sign, which requires the sign to be in maintenance mode.

Port Mgr Connect Timeout (sec)

Port Type ☐ ISDN ☐ POTS ☐ RS232 ☒ TCP/IP

Hostname / IP Addr

TCP Port

Port Managers N/A

CHART R3B3 Prototype 11/12/2008 © 2002-2008 MDSHA. All rights reserved.

Figure 5-81 Field Comm Settings - TCP/IP

When TCP/IP communications is selected, the user must enter the IP address and port used to communicate with the device. (Port managers do not apply to TCP/IP communications). After submitting the form, this data will appear on the details page for the device in the Comm Settings section (not shown) if the user has rights to view sensitive data for that device.

5.1.13.2 NTCIP DMS Font Settings

NTCIP DMS devices contain default font and default line spacing settings that are used when these settings are not specified within a message that is to be displayed on the device. CHART messages do not currently include font markup, which means the default font and line spacing settings configured on the device are used for all messages. On many of these signs, the default font and line spacing settings are not desirable and have been manually changed (as a setting for these defaults did not exist in CHART), however when power is cycled on the sign it reverts back to its original settings. A maintenance action is then required to reset the default font and line spacing settings for the sign. Until then messages are displayed in the undesirable (usually small) font.

In R3B3, settings are added for NTCIP DMSs that allow the desired settings for the sign's default font and line spacing to be stored in CHART. Each time CHART sets a message on the

sign it will set the sign's default font and line spacing to the values previously stored to ensure the settings are set correctly. CHART then displays the message as normal, and the desired font and line spacing will be used.

These new settings appear on the edit form used to edit the basic settings (shown below) and in the Basic Settings section of the DMS details page. They also appear on the form used to add an NTCIP DMS to the system in the General DMS Information section of that form.

The screenshot shows a web browser window titled "CHART - Windows Internet Explorer". The page header includes the "CHART" logo and navigation links for "Main Window" and "Help". The main heading is "Basic Settings For DMS: 4403". The form contains the following settings:

- Name: 4403
- Has Beacons: ☒
- Sign Type: char
- Owning Organization: SHA
- Enable Device Logging: ☐
- Display Size: Height (chars) 3, Width (chars) 21
- Max Pages: 2
- Character Size: 7x5 (H x W)
- Font: -- Select --
- Line Spacing: -- Select --
- Default Line Justification: center
- Default Page On Time: 2.5 sec
- Default Page Off Time: 0.0 sec
- Travel Time Msg Arb Queue Level: Travel Time
- Toll Rate Msg Arb Queue Level: Toll Rate

At the bottom of the form are "Submit" and "Cancel" buttons. Two red arrows point to the "Font" and "Line Spacing" dropdown menus.

Figure 5-82 NTCIP DMS Font and Line Spacing Settings

5.1.13.3 TSS Lane Level Detector Data

The detector details page, which previously showed traffic parameters only for zone groups, is enhanced in R3B3 to show the traffic parameters for each zone within a group, which usually equates to a single lane of traffic.

Traffic Parameters			
Zone Group	Speed (MPH)	Volume	Occupancy (%)
northbound	53	30	30%
↳ Zone 1	54 MPH	15	30%
↳ Zone 2	52 MPH	15	30%

Figure 5-83 Lane Level Detector Data

The Traffic Parameters section of the details page (shown above) will include a row for each zone of a zone group underneath a highlighted row for the zone group.

5.1.14 Public / Private Data Sharing

Most CHART user rights focus on the ability for users to perform certain actions, or to view whole classes of data (DMS data for example). R3B3 adds several fine grained user rights that are used to determine whether certain data elements can be viewed by users and in some cases how the data will be shown. The user interface screens where these new rights will be enforced are on the Traffic Events List, Traffic Event Details page, the Detector List, and Detector Details page. There is also a new/enhanced group of settings that control access to sensitive device configuration data, such as phone numbers and access codes. Details on user interface screens affected by these new data access user rights are discussed in the sections below.

5.1.14.1 Traffic Event List

A new user right will control the ability for the user to know if a collision includes a fatality.



Figure 5-84 Incident Name with Fatality

Users that do not possess this new user right will not see the word “Fatality” within the CHART System.

5.1.14.2 Traffic Event Details Page

The user right that determines the user’s ability to see if an incident is a fatality also applies to the traffic event details page. The event name will not contain the word Fatality if the user doesn’t have right to view sensitive incident data, and the incident type section of the form (shown below) will not show that the incident type is fatality.



Figure 5-85 Incident Details with Fatality

Another user right, View Traffic Event History, is used to control access to the traffic event history that is accessed via the traffic event details page. This right can be granted for all organizations, or for individual organizations. If the traffic event is not owned by an organization for which this user right has been granted, the user will not have access to the traffic event history link on the traffic event details page.



Figure 5-86 Event History Link

5.1.14.3 Detector List

Two new user rights will be used to determine if a user can view the speed data for a detector and if so, whether they can see a summary or the actual speed. Both of these new rights can be granted for all organizations or just for specific organizations. Users without either right for the owning organization of a detector will not see any speed data for that detector in the list. Users with the right to view summary data for the detector's owning organization will see a speed range for the detector, and users with the right to view detailed data for the detector's owning organization will see the actual speed reading.

 Rich RTMS 2 Rich's Office		30-50 MPH
 Rich RTMS 3 Rich's Office		38 MPH

Figure 5-87 Summary and Detailed Speed Data

5.1.14.4 Detector Details Page

The detector details page will enforce the same rights as discussed in the section above, however they will also control access to the zone group and zone level data. A user without either of the rights that allow detector data to be viewed will not see any speed, volume, or occupancy on the details page. Users that have the right to view summary data for the detector's owning organization will be able to see speed ranges for zone groups and zones, but no volume or occupancy. Users that have the right to view detailed data for the detector's owning organization will be able to see actual volume, speed, and occupancy data for the detector.

Traffic Parameters

Zone Group	Speed (MPH)	Volume	Occupancy (%)
northbound	53	30	30%
↳ Zone 1	54 MPH	15	30%
↳ Zone 2	52 MPH	15	30%

Figure 5-88 Traffic Parameters showing Detailed Data

5.1.14.5 Sensitive Device Configuration Data

Prior to R3B3, several user rights existed that could be granted to allow users to view configuration data for various devices. These rights were inconsistent across device types, and for at least one device type non-existent, allowing any user that can view the list of devices to see the configuration details for the device. R3B3 adds new user rights that will be required to view the sensitive configuration data for a device. There is one new “view sensitive config” right for each device type. These new user rights are enforced on the details page for the associated device. Users that do not possess the right for the device type and the device’s owning organization will not be permitted to view sensitive configuration information such as phone numbers and access codes. The image below shows a sample of sensitive device configuration information that requires a special right to view in R3B3.

Comm Settings:

Drop Address:	1
Port Mgr Connect Timeout:	15 sec
Port Type:	ISDN Modem
Baud:	9600
Data Bits:	8
Parity:	None
Stop Bits:	1
Flow Control:	None
Default Phone Number:	(301) 555-0965
Port Managers:	

Name	Device	Phone Number
swsim		(301) 555-0965

Figure 5-89 Sensitive Device Configuration Data

5.2 Alert Module

5.2.1 Classes

5.2.1.1 AlertModule (Class Diagram)

This class diagram defined the classes in the AlertModule package. These classes define the AlertModule server. It utilizes generated IDL classes as wells as other Chart2 utility classes.

periodically reviews the alerts in the delay state for those whose delay period has expired. As with the accept state timeout, the delay timeout period is established in the system profile for each alert type. When either the accept timeout or the delay timeout expires, this task calls into the AlertImpl to escalate the alert.

5.2.1.1.2 Alert (Class)

This is a CORBA interface that provides access to information pertaining to an Alert and provides operations used to manage an alert.

5.2.1.1.3 AlertData (Class)

This is a CORBA struct, defined in IDL, that contains the data that applies to all alert types.

5.2.1.1.4 AlertDB (Class)

This class provides a database interface for the AlertModule. It includes methods needed to store and retrieve Alert related information.

5.2.1.1.5 AlertFactory (Class)

This IDL interface contains the operations available for an Alert Factory. The AlertFactory is responsible for creating alerts and storing alert information on the alerts that it created.

5.2.1.1.6 AlertFactoryImpl (Class)

This AlertFactoryImpl class implements the IDL AlertFactory interface and is responsible for creating and managing the objects created to represent alerts (AlertImpls) in the Chart2 system.

5.2.1.1.7 AlertImpl (Class)

The AlertImpl class implements the IDL Alert interface. The AlertImpl class contains the base class functionality for all other alert types in the Chart2 system. Each instance of one of the AlertImpls derived types represents a specific alert.

5.2.1.1.8 AlertModule (Class)

This class provides the resources and support functionality necessary to serve alert related objects in a service application. It implements the ServiceApplicationModule interface which allows it to be served from any ServiceApplication.

5.2.1.1.9 AlertModuleProperties (Class)

This class provides operations for getting values in the service's java properties file.

5.2.1.1.10 AlertPrivateData (Class)

This class contains base alert data which is private to the AlertImpl class. Among the data stored in AlertPrivateData is the time of the previous escalation or reset time, and the

isOffline flag to indicate the alert is ready for archiving.

5.2.1.1.11 ArchiveTimerTask (Class)

This class implements the alert archive timer task. It periodically sweeps through the closed alerts in the system for those alerts deemed old enough to be archived. If an alert is found that has aged beyond the system defined archive timer limit, it will set a flag on the alert to mark it for removal. At some later time a separate database task will run to remove and off-load these alerts to an archive file.

5.2.1.1.12 DataModel (Class)

The data model class serves as a collection of objects. It provides an efficient lookup mechanism for locating any object, and methods which allow for the retrieval of all objects of a particular type. Additionally, this class provides the ability to attach observer objects which are notified when objects are added to or removed from the model. Objects may also notify the DataModel that they have been modified. The model will periodically notify all attached observers of the changes to objects in the model.

5.2.1.1.13 DBConnectionManager (Class)

This class implements a database connection manager that manages a pool of database connections. Any CHART II system thread requiring database access gets a database connection from the pool of connections maintained by this manager class. The connections are maintained in two separate lists namely, inUseList and freeList. The inUseList contains connections that have already been assigned to a thread. The freeList contains unassigned connections. This class assumes that an appropriate JDBC driver has been loaded either by using the "jdbc.drivers" system property or by loading it explicitly. The class has a monitor thread that is started by the constructor. This connection monitor thread periodically checks the inuseList to see if there are connections that are owned by dead threads and move such connections to the freeList. The connection monitor thread is started only if a non-zero value is specified for the monitoring time interval in the constructor.

5.2.1.1.14 DeviceFailureAlert (Class)

This IDL interface contains operations specific to a Device Failure alert. This interface is implemented by classes representing DeviceFailureAlerts in the Chart2 System.

5.2.1.1.15 DeviceFailureAlertData (Class)

This is a CORBA struct, defined in IDL, that contains base alert data plus data specific to a DeviceFailureAlert. Specific to this alert is the traffic event id of the failed device event causing the alert. Also included is information on the device failure type.

5.2.1.1.16 DeviceFailureAlertImpl (Class)

The DeviceFailureAlertImpl class is derived from the AlertImpl class and implements the IDL DeviceFailureAlert interface. Type specific functionality is provided by this class for

Device Failure alerts.

5.2.1.1.17 DuplicateEventAlert (Class)

This IDL interface contains operations specific to a Duplicate Event alert. This interface is implemented by classes representing DuplicateEventAlertsDevice in the Chart2 System.

5.2.1.1.18 DuplicateEventAlertData (Class)

This is a CORBA struct, defined in IDL, that contains base alert data plus data specific to a DuplicateEventAlert. Specific to this alert are the event ids of the two probable duplicate traffic events.

5.2.1.1.19 DuplicateEventAlertImpl (Class)

The DuplicateEventAlertImpl class is derived from the AlertImpl class and implements the IDL DuplcateEventAlert interface. Type specific functionality is provided by this class for Duplicate Event alerts.

5.2.1.1.20 EscalateTimerTask (Class)

This class implements the alert escalate timer task. It periodically checks the new alerts in the system for those that have not been accepted, delayed, or closed within the escalation timeout period. This timeout period is established in the system profile for each alert type. If an alert is found that has exceeded the escalation timer limit, a call into AlertImpl will be made to escalate the alert.

5.2.1.1.21 EventStillOpenAlert (Class)

This IDL interface contains operations specific to a Event Still Open alert. This interface is implemented by classes representing EventStillOpenAlerts in the Chart2 System.

5.2.1.1.22 EventStillOpenAlertData (Class)

This is a CORBA struct, defined in IDL, that contain the base alert data plus data specific to an EventStillOpenAlert. Specific to this alert is the id of the traffic event that is still open.

5.2.1.1.23 EventStillOpenAlertImpl (Class)

The EventStillOpenAlertImpl class is derived from the AlertImpl class and implements the IDL EventStillOpenAlert interface. Type specific functionality is provided by this class for Event Still Open alerts.

5.2.1.1.24 ExecuteScheduledActionsAlert (Class)

This IDL interface contains operations specific to aExecute Scheduled Actions alert. This interface is implemented by classes representing ExecuteScheduledActionsAlert in the Chart2 System.

5.2.1.1.25 ExecuteScheduledActionsAlertData (Class)

This is a CORBA struct, defined in IDL, that contains the base alert data plus data specific to an ExecuteScheduledActionsAlert.

5.2.1.1.26 ExecuteScheduledActionsAlertImpl (Class)

The ExecuteScheduledEventAlertImpl class is derived from the AlertImpl class and implements the IDL ExecuteScheduledEventAlert interface. Type specific functionality is provided by this class for scheduled event alerts.

5.2.1.1.27 ExternalConnectionAlert (Class)

This IDL interface contains operations specific to an External Connection Alert, which indicates trouble with a connection between CHART and an external system.

5.2.1.1.28 ExternalConnectionAlertData (Class)

This IDL structure contains data specific to an External Connection Alert, e.g., the ID of the interface which is having trouble and a flag indicating whether the connection is in failure or warning status, the timestamp it transitioned. (The GUI displays additional data which is best acquired from the GUI's object cache.) (Text in the base AlertData structure provides a textual description and alert management data.)

5.2.1.1.29 ExternalConnectionAlertImpl (Class)

This is the implementation of the External Connection Alert, which alerts users to trouble with an external connection. This can be a failure or a warning status. (Users can specify whether to receive failures and warnings, or just failures).

5.2.1.1.30 ExternalEventAlert (Class)

This IDL interface contains operations specific to an External Event Alert, which indicates an event has arrived from an external system which satisfies criteria a CHART administrator has defined to flag an external event as significant enough to warrant this alert.

5.2.1.1.31 ExternalEventAlertData (Class)

This IDL structure contains data specific to an External Event Alert, e.g., the ID of the event and the ID of the first rule found that requested an alert be sent. (Text in the base AlertData structure provides a textual description and alert management data.)

5.2.1.1.32 ExternalEventAlertImpl (Class)

This is the implementation of the External Event Alert, triggered by receipt of events with match the external event alert settings in the event import module, as defined by a CHART administrator.

5.2.1.1.33 GenericAlert (Class)

This IDL interface contains operations specific to a Generic alert. This interface is implemented by classes representing GenericAlerts in the Chart2 System.

5.2.1.1.34 GenericAlertImpl (Class)

The GenericAlertImpl class is derived from the AlertImpl class and implements the IDL GenericAlert interface. Any type specific functionality that may be implemented in the future would be provided by this class for Generic alerts.

5.2.1.1.35 java.util.Properties (Class)

The Properties class represents a persistent set of properties. The Properties can be saved to a stream or loaded from a stream. Each key and its corresponding value in the property list is a string. A property list can contain another property list as its "defaults"; this second property list is searched if the property key is not found in the original property list.

5.2.1.1.36 java.util.Timer (Class)

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

5.2.1.1.37 java.util.TimerTask (Class)

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

5.2.1.1.38 ObjectCache (Class)

The ObjectCache is a wrapper for the DataModel. It provides access to DataModel methods to find objects in the data model, delegating those methods to the DataModel itself. It also provides additional methods of finding name filtered objects and discovering "duplicate" objects (as defined by an isDuplicateOf() method of the Duplicatable interface).

5.2.1.1.39 ProxyAlert (Class)

This class is used as a proxy for alerts existing in all alert modules in the system (including the local service). The complete set of data for each alert is stored in the ProxyAlert, along with its ID and a reference to the Alert object it represents. These proxy alerts allow every alert module service in the system to have some knowledge of every alert in the entire system, for the quickly determining whether a proposed new alert already exists elsewhere in the alert system (and therefore does not need to be redundantly entered into the system again). ProxyAlert implements the Duplicatable interface, so that the ObjectCache can generically be queried to check for duplicates of any other ProxyAlert. This ProxyAlert class is the super class for derived classes for each specialized type of alert in the system, so that type specific data can be stored and accessed for each alert type, and can be queried for comparison for the Duplicatable isDuplicateOf() method.

5.2.1.1.40 PushEventConsumer (Class)

This class is a utility class which will be responsible for connecting a consumer implementation to an event channel, and maintaining that connection. When the `verifyConnection` method is called, this object will determine if the channel has been lost and will attempt to re-connect to the channel if it has.

5.2.1.1.41 PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.2.1.1.42 ServiceApplication (Class)

This interface is implemented by objects that can provide the basic services needed by a ChartII service application. These services include providing access to basic CORBA objects that are needed by service applications, such as the ORB, POA, Trader, and Event Service.

5.2.1.1.43 ServiceApplicationModule (Class)

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

5.2.1.1.44 TollRateAlert (Class)

This IDL interface contains operations specific to an Toll Rate Alert, which indicates a travel route which had a currently active toll rate no longer does in a more recently received toll rate update document from a toll rate provider. (This alert is not sent if a toll rate expires due to an absence of any current toll rate document -- such an event would have triggered one external connection alert and does not need to also trigger a multitude of individual toll rate alerts as well.)

5.2.1.1.45 TollRateAlertData (Class)

This IDL structure contain data specific to a Toll Rate Alert, e.g., the travel route which no longer has data for its toll rate. (Text in the base AlertData structure provides a textual description and alert management data.)

5.2.1.1.46 TollRateAlertImpl (Class)

This is the implementation of the Toll Rate Alert, which is sent when a toll rate document is received from the toll rate supplier which is missing a toll rate which had been present in the prior document.

5.2.1.1.47 TravelTimeAlert (Class)

This IDL interface contains operations specific to an Travel Time Alert, which indicates the travel time associated with a travel route is high enough to warrant this alert.

5.2.1.1.48 TravelTimeAlertData (Class)

This IDL structure contains data specific to a Travel Time Alert, e.g., the travel time limit and the travel time which exceeded the limit. (Text in the base AlertData structure provides a textual description and alert management data.)

5.2.1.1.49 TravelTimeAlertImpl (Class)

This is the implementation of the Travel Time Alert, which is sent when the travel time calculated for a Travel Route exceeds the alert threshold configured for the Travel Route by a CHART Administrator.

5.2.1.1.50 UnhandledResourcesAlert (Class)

This IDL interface contains operations specific to a Unhandled Resources alert. This interface is implemented by classes representing UnhandledResourceAlerts in the Chart2 System.

5.2.1.1.51 UnhandledResourcesAlertData (Class)

This is a CORBA struct, defined in IDL, that contains the base alert data plus data specific to an UnhandledResourcesAlert.

5.2.1.1.52 UnhandledResourcesAlertImpl (Class)

The UnhandledResourceAlertImpl class is derived from the AlertImpl class and implements the IDL UnhandledResourceAlert interface. Type specific functionality is provided by this class for Unhandled Resource alerts.

5.2.1.1.53 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.2.1.2 ProxyAlertClasses (Class Diagram)

This class diagram shows all classes related to the storage of proxy alerts in the object cache. The ProxyAlert class, and its subclasses, provide access to all alerts known to be in the system, so that an alert factory can quickly determine whether a requested new alert already exists elsewhere in the alert system (and therefore does not need to be redundantly entered into the system again).

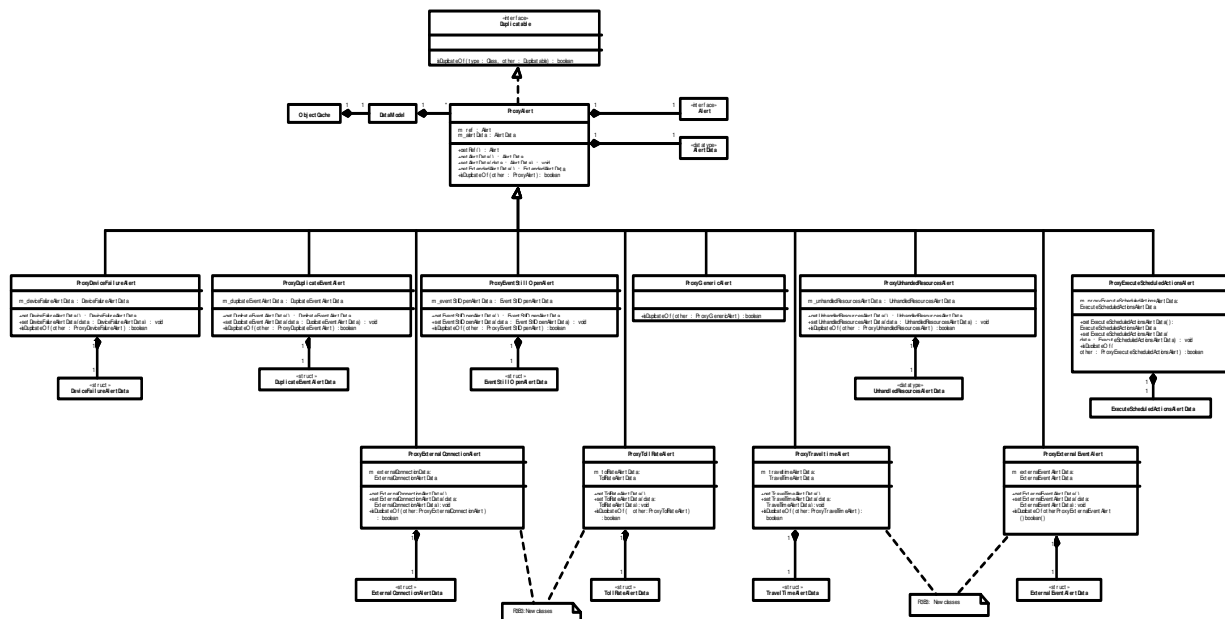


Figure 5-91 ProxyAlertClasses (Class Diagram)

5.2.1.2.1 Alert (Class)

This is a CORBA interface that provides access to information pertaining to an Alert and provides operations used to manage an alert.

5.2.1.2.2 AlertData (Class)

This is a CORBA struct, defined in IDL, that contains the data that applies to all alert types.

5.2.1.2.3 DataModel (Class)

The data model class serves as a collection of objects. It provides an efficient lookup mechanism for locating any object, and methods which allow for the retrieval of all objects of a particular type. Additionally, this class provides the ability to attach observer objects which are notified when objects are added to or removed from the model. Objects may also notify the DataModel that they have been modified. The model will periodically notify all attached observers of the changes to objects in the model.

5.2.1.2.4 DeviceFailureAlertData (Class)

This is a CORBA struct, defined in IDL, that contains base alert data plus data specific to a DeviceFailureAlert. Specific to this alert is the traffic event id of the failed device event causing the alert. Also included is information on the device failure type.

5.2.1.2.5 Duplicatable (Class)

This java interface is implemented by classes which have sense of being "duplicated" within the CHART system. This allows the ObjectCache to search for duplicates of any Duplicatable object. This is different from "equals()" or "compareTo()". To cite two examples: Alerts within CHART are duplicates if they refer to the same objects within CHART (but do not have the same Alert ID, which is more closely associated with "equals()"). Traffic Events within CHART are duplicates if they have the same location (but do not have the same Traffic Event ID).

5.2.1.2.6 DuplicateEventAlertData (Class)

This is a CORBA struct, defined in IDL, that contains base alert data plus data specific to a DuplicateEventAlert. Specific to this alert are the event ids of the two probable duplicate traffic events.

5.2.1.2.7 EventStillOpenAlertData (Class)

This is a CORBA struct, defined in IDL, that contain the base alert data plus data specific to an EventStillOpenAlert. Specific to this alert is the id of the traffic event that is still open.

5.2.1.2.8 ExecuteScheduledActionsAlertData (Class)

This is a CORBA struct, defined in IDL, that contains the base alert data plus data specific to an ExecuteScheduledActionsAlert.

5.2.1.2.9 ExternalConnectionAlertData (Class)

This IDL structure contains data specific to an External Connection Alert, e.g., the ID of the interface which is having trouble and a flag indicating whether the connection is in failure or warning status, the timestamp it transitioned. (The GUI displays additional data which is best acquired from the GUI's object cache.) (Text in the base AlertData structure provides a textual description and alert management data.)

5.2.1.2.10 ExternalEventAlertData (Class)

This IDL structure contains data specific to an External Event Alert, e.g., the ID of the event and the ID of the first rule found that requested an alert be sent. (Text in the base AlertData structure provides a textual description and alert management data.)

5.2.1.2.11 ObjectCache (Class)

The ObjectCache is a wrapper for the DataModel. It provides access to DataModel

methods to find objects in the data model, delegating those methods to the DataModel itself. It also provides additional methods of finding name filtered objects and discovering "duplicate" objects (as defined by an isDuplicateOf() method of the Duplicatable interface).

5.2.1.2.12 ProxyAlert (Class)

This class is used as a proxy for alerts existing in all alert modules in the system (including the local service). The complete set of data for each alert is stored in the ProxyAlert, along with its ID and a reference to the Alert object it represents. These proxy alerts allow every alert module service in the system to have some knowledge of every alert in the entire system, for the quickly determining whether a proposed new alert already exists elsewhere in the alert system (and therefore does not need to be redundantly entered into the system again). ProxyAlert implements the Duplicatable interface, so that the ObjectCache can generically be queried to check for duplicates of any other ProxyAlert. This ProxyAlert class is the super class for derived classes for each specialized type of alert in the system, so that type specific data can be stored and accessed for each alert type, and can be queried for comparison for the Duplicatable isDuplicateOf() method.

5.2.1.2.13 ProxyDeviceFailureAlert (Class)

This subclass of ProxyAlert is used to cache DeviceFailureAlert types of alerts. It holds and provides access to data specific to the DeviceFailureAlert, and provides an isDuplicateOf() implementation specialized for comparing two alerts of this type.

5.2.1.2.14 ProxyDuplicateEventAlert (Class)

This subclass of ProxyAlert is used to cache DuplicateEventAlert types of alerts. It holds and provides access to data specific to the DuplicateEventAlert, and provides an isDuplicateOf() implementation specialized for comparing two alerts of this type.

5.2.1.2.15 ProxyEventStillOpenAlert (Class)

This subclass of ProxyAlert is used to cache EventStillOpenAlert types of alerts. It holds and provides access to data specific to the EventStillOpenAlert, and provides an isDuplicateOf() implementation specialized for comparing two alerts of this type.

5.2.1.2.16 ProxyExecuteScheduledActionsAlert (Class)

This subclass of ProxyAlert is used to cache ExecuteScheduledActionsAlert types of alerts. It holds and provides access to data specific to the ExecuteScheduledActionsAlert, and provides an isDuplicateOf() implementation specialized for comparing two alerts of this type.

5.2.1.2.17 ProxyExternalConnectionAlert (Class)

This class is used to carry data about an external connection alert which has been received by the ObjectCache from an AlertModule. Proxy alerts are collected and used by all Alert Modules to aid in de-duping alerts which could be created via multiple Alert Modules.

5.2.1.2.18 ProxyExternalEventAlert (Class)

This class is used to carry data about an external event alert which has been received by the ObjectCache from an AlertModule. Proxy alerts are collected and used by all Alert Modules to aid in de-duping alerts which could be created via multiple Alert Modules.

5.2.1.2.19 ProxyGenericAlert (Class)

This subclass of ProxyAlert is used to cache GenericAlert types of alerts. It holds and provides access to data specific to the GenericAlert, and provides an isDuplicateOf() implementation specialized for comparing two alerts of this type.

5.2.1.2.20 ProxyTollRateAlert (Class)

This class is used to carry data about a toll rate alert which has been received by the ObjectCache from an AlertModule. Proxy alerts are collected and used by all Alert Modules to aid in de-duping alerts which could be created via multiple Alert Modules.

5.2.1.2.21 ProxyTraveltimeAlert (Class)

This class is used to carry data about a travel time alert which has been received by the ObjectCache from an AlertModule. Proxy alerts are collected and used by all Alert Modules to aid in de-duping alerts which could be created via multiple Alert Modules.

5.2.1.2.22 ProxyUnhandedResourcesAlert (Class)

This subclass of ProxyAlert is used to cache UnhandledResourcesAlert types of alerts. It holds and provides access to data specific to the UnhandledResourcesAlert, and provides an isDuplicateOf() implementation specialized for comparing two alerts of this type.

5.2.1.2.23 TollRateAlertData (Class)

This IDL structure contain data specific to a Toll Rate Alert, e.g., the travel route which no longer has data for its toll rate. (Text in the base AlertData structure provides a textual description and alert management data.)

5.2.1.2.24 TravelTimeAlertData (Class)

This IDL structure contains data specific to a Travel Time Alert, e.g., the travel time limit and the travel time which exceeded the limit. (Text in the base AlertData structure provides a textual description and alert management data.)

5.2.1.2.25 UnhandledResourcesAlertData (Class)

This is a CORBA struct, defined in IDL, that contains the base alert data plus data specific

to an UnhandledResourcesAlert.

5.3 Camera Control Module

5.3.1 Classes

5.3.1.1 VideoHighLevel (Class Diagram)

This diagram shows the High Level CHART II CORBA interfaces. This diagram does not show all VideoService IDL elements, but shows the highest level elements and their interrelationships. For further details, see VideoHighLevel-VideoSource, VideoHighLevel-VideoSink, and VideoHighLevel-VideoTransmission diagrams. The collection of these last three diagrams show all planned CORBA/IDL interface objects for the CHART II Video Service. In all four of these diagrams, some boxes are shown indicating objects planned to be implemented for later releases. These objects have been considered for future planning purposes, to ensure that the current design is well-thought out enough to be able to accommodate future planned enhancements.

This diagram shows cameras and related information generally on the left side, monitors and related information generally on the right side, and video transmission and routing capabilities in the central part of the diagram. The VideoProvider interface is the top of the interface set which contains the VideoCamera interface. VideoSource includes video sources including fixed cameras, image generators, etc. Likewise on the right side, VideoCollector is at the top, opposite VideoProvider, with VideoSink and Monitor lower down. In addition to VideoSource and VideoSink objects, BridgeCircuit objects will also be VideoProviders and VideoCollectors, since any bridge circuit both collects video from some other VideoProvider and provides video to the next VideoCollector in line. Multiple bridge circuits may be present between the ultimate VideoProvider (i.e., the VideoSource, that is, the camera, the true source of the image) and the ultimate VideoCollector (i.e., the VideoSink, that is, the monitor, the final sink of the image).

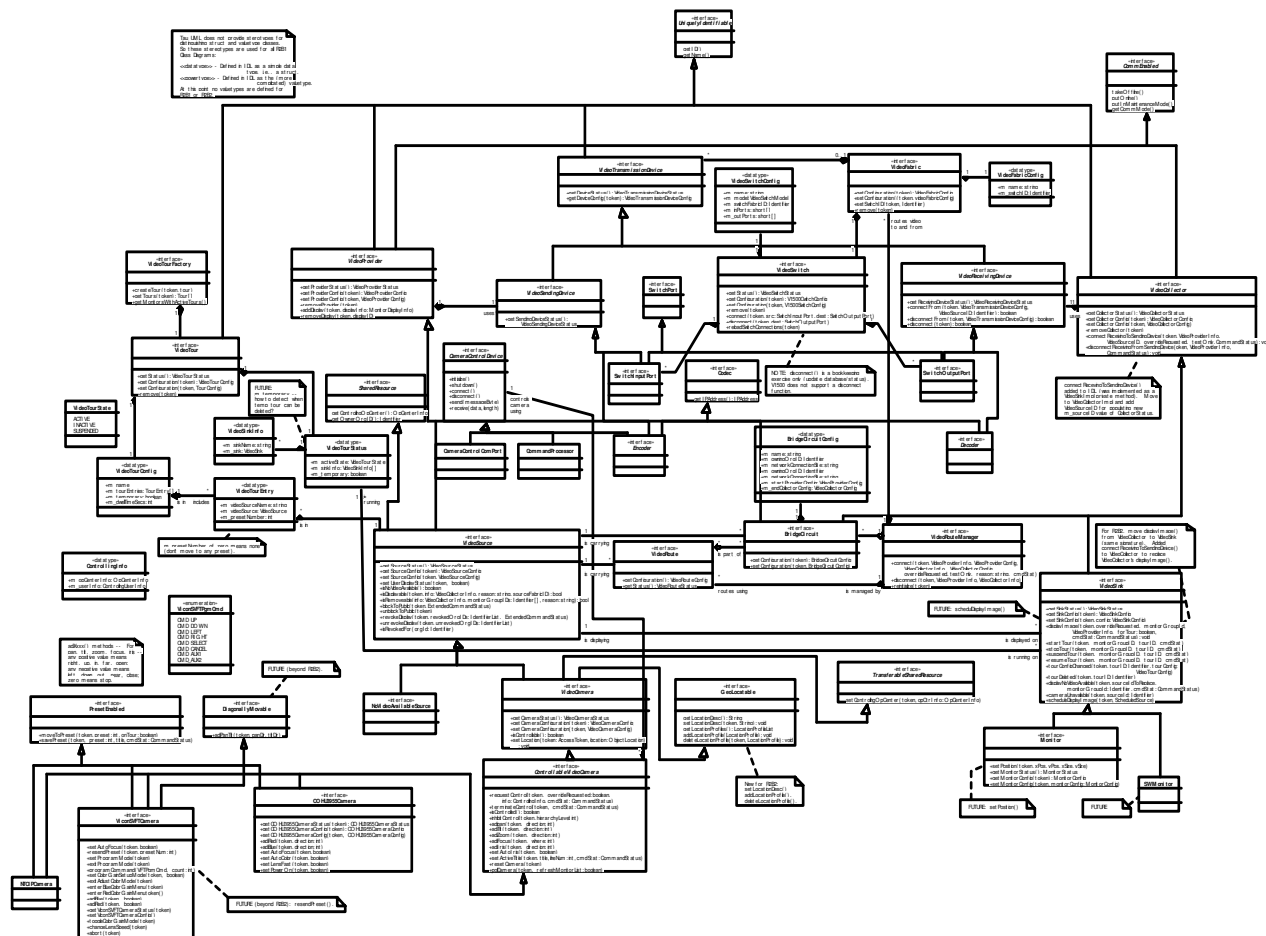


Figure 5-92 VideoHighLevel (Class Diagram)

5.3.1.1.1 BridgeCircuit (Class)

The BridgeCircuit interface is implemented by a objects which serve to bridge disparate switch fabrics within video routes. These switch fabrics would include the switch fabrics based around a V1500 switch and also the "null" switch fabric consisting of no switch and codec VideoTransmissionDevice objects. The BridgeCircuit interface includes both the VideoCollector interface (meaning the BridgeCircuit receives video from another VideoProvider, ultimately the VideoSource) and the VideoProvider interface (meaning the BridgeCircuit provides video to another VideoCollector, ultimately to one or more VideoSink objects).

5.3.1.1.2 BridgeCircuitConfig (Class)

This represents configuration information for a bridge circuit. This is the status of a BridgeCircuit object. It consists primarily of configuration of the VideoProvider side (input to the bridge circuit) and of the VideoCollector side (output of the bridge circuit).

5.3.1.1.3 CameraControlComPort (Class)

The CameraControlComPort interface is implemented by a class representing a COM port with direct connection to the control port of a video camera. It is used to send video camera control commands and return responses to a camera control process.

5.3.1.1.4 CameraControlDevice (Class)

The CameraControlDevice interface is implemented by classes which provides communications for access to control functions for a video camera. This includes encoders and direct COM ports.

5.3.1.1.5 Codec (Class)

The Codec interface is implemented by objects representing codec devices (that is, encoders and decoders). It defines generic methods to be implemented by both encoders and decoders.

5.3.1.1.6 COHU3955Camera (Class)

The COHU3955Camera interface is implemented by objects representing COHU model 3055 video cameras. It extends the ControllableVideoCamera interface by adding methods unique to the COHU 3955 cameras (unique within the universe of camera types planned for implementation within CHART II).

5.3.1.1.7 CommandProcessor (Class)

The CommandProcessor class provides an implementation of the CommandProcessor interface and is derived from the CameraControlDevice class. The CommandProcessor manages the control of multiple cameras attached to one or more COM ports. The CommandProcessor may or may not be local to the camera that is being controlled.

5.3.1.1.8 CommEnabled (Class)

The CommEnabled interface is implemented by objects that can have their communications turned on or off. This interface also supports a maintenance mode (although any given implementation may choose to implement putInMaintenanceMode() by throwing a CHART2Exception, if maintenance mode is not supported by that particular implementation). This interface is typically implemented only for field devices.

5.3.1.1.9 ControllableVideoCamera (Class)

The ControllableVideoCamera interface is implemented by objects representing controllable video cameras within the CHART II system. The ControllableVideoCamera interface represents a controllable video camera as opposed to an uncontrollable, immovable VideoCamera. Current plans call for classes to represent a COHU 3955 camera, Vicon SVFT camera, and NTCIP-compliant camera, and there are interfaces defined for each of these subtypes of ControllableVideoCamera. The ControllableVideoCamera interface includes all methods common to the two known types

of video cameras currently in use by MDSHA, although it is likely to contain a superset of methods which would be implemented by the entire universe of all video cameras which could someday be used. This interface may have to be refined in the event that future brands or models of video cameras might be incorporated under CHART II, but it is an appropriate set of methods for the present day. Current plans call for classes to represent a COHU 3955 camera, Vicon SVFT camera, and NTCIP-compliant camera.

5.3.1.1.10 ControllingInfo (Class)

The ControllingInfo structure contains information about the entity controlling (or requesting to control) a VideoCamera.

5.3.1.1.11 Decoder (Class)

The Decoder interface is implemented by classes representing any type of video decoder. The Decoder interface includes both the Codec and the VideoReceivingDevice interfaces.

5.3.1.1.12 DiagonallyMovable (Class)

The DiagonallyMovable interface is implemented by VideoCamera-enabled classes which can be moved diagonally in addition to standard orthogonal pan and tilt commands. A particular implementation may support 45-degree movements only, in which case the panDir and tiltDir parameters are +/- 1 to indicate direction only, or an implementation may support 360 degrees of motion, in which case, in addition to signs, the relative ratios of the parameters indicate the percent of movement proportionally in the pan/tilt directions. This interface is expected to be implemented beyond R2B2.

5.3.1.1.13 Encoder (Class)

The Encoder interface is implemented by classes representing any type of video encoder. The Encoder interface includes both the Codec and the VideoSendingDevice interfaces, which means in addition to providing forwarding of video, it also is used to send video camera control commands and return responses to a camera control process.

5.3.1.1.14 Monitor (Class)

The Monitor interface is implemented by objects which represent a video monitor, e.g., a real, physical "television set" on which a video image can be displayed. This is the most common type of VideoSink (the other being a SWMonitor, part of a future requirement to stream video directly to user's workstations (or potentially other nearby computers)).

5.3.1.1.15 NoVideoAvailableSource (Class)

The FixedVideoSource interface is implemented by objects which represent a video source other than a video camera, such as the "No Image Available" image generators. This interface could also represent a VCR or any other video source that is not a camera. The FixedVideoSource does not include the GeoLocatable interface because the location (e.g. lat/long) of a fixed video source is irrelevant in CHART II processing (unlike for a

VideoCamera, for which the location (lat/long) of a camera could someday be used for automatic identification of cameras near traffic events, automatic pointing of cameras to traffic events, etc.)

5.3.1.1.16 NTCIPCamera (Class)

The NTCIPCamera interface is implemented by objects which support the NTCIP standard for CCTV cameras. As this is a future requirement for cameras not currently fielded by MDSHA, this interface is left to be defined at a later time.

5.3.1.1.17 PresetEnabled (Class)

The PresetEnabled interface is implemented by VideoCamera-enabled classes which can store and move to presets. The savePreset() method saves the current camera position as the preset position. This interface is expected to be implemented in R2B2.

5.3.1.1.18 SharedResource (Class)

The SharedResource interface is implemented by any object that must always have an operations center responsible for the disposition of the resource while the resource is in use.

5.3.1.1.19 SwitchInputPort (Class)

This is the interface for a switch input port. A switch input port is a type of switch port and is also a type of VideoSendingDevice, meaning it can send a video signal on behalf of the VideoProvider attached to it to any one or more VideoReceivingDevices (and corresponding VideoCollectors).

5.3.1.1.20 SwitchOutputPort (Class)

This is the interface for a switch output port. A switch output port is a type of switch port and is also a type of VideoReceivingDevice (meaning it receives a video signal on behalf of the VideoCollector attached to it). As VideoReceivingDevice, a SwitchOutputPort is capable of being connected to any VideoSendingDevice.

5.3.1.1.21 SwitchPort (Class)

There is a generic SwitchPort interface. It is a CommEnabled interface, meaning a SwitchPort can be online or offline. (A SwitchPort cannot be in maintenance mode).

5.3.1.1.22 SWMonitor (Class)

The SWMonitor interface is implemented by objects which represent a software monitor capable of receiving and displaying video (i.e., a streaming video MPEG software decoder running on a PC). This interface supports a future requirement to display video directly to user's workstations.

5.3.1.1.23 TransferableSharedResource (Class)

The TransferrableSharedResource interface is implemented by any object that must always have an operations center responsible for the disposition of the resource while the resource is in use but may also be allowed to transfer control of that resource to another operations center.

5.3.1.1.24 UniquelyIdentifiable (Class)

The UniquelyIdentifiable interface is implemented by classes whose instances have a unique identifier that is guaranteed not to match the identifier of any other uniquely identifiable objects in the system. It provides access to the unique ID, and the name (which does not have to be unique).

5.3.1.1.25 ViconSVFTCamera (Class)

The ViconSVFTCamera interface is implemented by a class representing the Vicon Surveyor VFT model video camera. (As there are no other Vicon brand cameras used within CHART II, there is no base ViconCamera interface representing all Vicon-brand cameras. For one thing, there would be no known basis for allocating methods to the base interface and the VFT interface.)

5.3.1.1.26 ViconSVFTPgmCmd (Class)

The ViconSVFTPgmCmd enumeration defines the values that can be used in the programCommand() method of the ViconSVFTCamera interface.

5.3.1.1.27 VideoCamera (Class)

The VideoCamera interface is implemented by objects representing video cameras within the CHART II system. Classes implementing this interface (and nothing below this interface would be fixed (non-controllable) video cameras. The VideoCamera interface includes the GeoLocatable interface, to someday allow for advanced features such as automatic identification of cameras near traffic events, automatic pointing of cameras to traffic events, etc.

5.3.1.1.28 VideoCollector (Class)

The VideoCollector interface is a generic abstract interface including VideoSink objects (e.g. video monitors) and BridgeCircuit objects. Both VideoSink and BridgeCircuit objects collect video from a VideoProvider, but only VideoSink objects are true destination endpoints for video feeds which a typical user would have direct interaction with. BridgeCircuit VideoCollector objects are merely an intermediate step in a VideoRoute which eventually provides video ultimately to the VideoSink object(s) at the end of the route(s).

5.3.1.1.29 VideoFabric (Class)

The VideoFabric is implemented by a class which represents a "video fabric", that is a

collection of VideoTransmissionDevice objects on a common "fabric" across which video can be created directly. This includes any collection of switch input ports and switch output ports on a single video switch. (Note that a collection of encoder and decoder types of VideoTransmissionDevice objects represents a different video fabric, across which video can be routed directly. The IP encoder/decoder fabric therefore is different from other fabrics in that it has no associated video switch.

5.3.1.1.30 VideoFabricConfig (Class)

The VideoFabricConfig structure is used to store and transmit configuration information about a VideoFabric object.

5.3.1.1.31 VideoProvider (Class)

The VideoProvider interface is a generic abstract interface including VideoSource objects (e.g. video cameras) and BridgeCircuit objects. Both VideoSource and BridgeCircuit objects provide video to a VideoCollector, but only VideoSource objects are true origins of video which a typical user would have direct interaction with. BridgeCircuit VideoProvider objects merely pass on video provided from elsewhere in a VideoRoute.

5.3.1.1.32 VideoReceivingDevice (Class)

The VideoReceivingDevice interface is implemented by objects which can be used to receive video from a corresponding VideoSendingDevice. A VideoReceivingDevice may be an MPEG decoder or may be an output port on a video switch.

5.3.1.1.33 VideoRoute (Class)

This interface defines a route through CHART II video distribution system. A given implementation of a VideoRoute may or may not be actively in use at any given time. Routes are defined by the combinations of all bridge circuits between all pairs of switch fabrics within the CHART II video distribution system. Routes cannot be added or deleted or enabled or disabled by users explicitly: the routes and their status are defined implicitly by the configuration and status of bridge circuits defined in the system at any given time.

5.3.1.1.34 VideoRouteManager (Class)

The VideoRouteManager interface is implemented by a class which provides video routing capabilities within CHART II. This router does not need to be used (in fact, cannot be used) when the VideoSource and VideoSink are on the same switch fabric -- it is used only to make video routes across switch fabrics. The implementation will use a set of rules to arbitrate among requested video displays when a set of bridge circuits between one or more pairs of switch fabrics is fully utilized. Based on the override rules implemented, a new incoming routing request may or may not be able to be fulfilled depending upon priority, routing guarantees, number of images viewed, ongoing camera control sessions, etc. If an override can be granted, the overridden route(s) will be dropped in favor of the new route.

5.3.1.1.35 VideoSendingDevice (Class)

The VideoSendingDevice interface is implemented by objects which can be used to send video to a corresponding VideoReceivingDevice. A VideoSendingDevice may be an MPEG encoder or may be an input port on a video switch.

5.3.1.1.36 VideoSink (Class)

The VideoSink interface is implemented by objects which serve as final endpoints for video signals, such as video monitors and streaming video receivers directly on user workstations. Within the user interface, the VideoSink interface represents all objects on which a video source can be placed for viewing by users.

5.3.1.1.37 VideoSinkInfo (Class)

VideoSinkInfo represent information about a VideoSink that is used by a VideoTour.

5.3.1.1.38 VideoSource (Class)

The VideoSource interface is implemented by objects which originate video signals, such as video cameras and image generators. Within the user interface, the VideoSource interface represents all video sources which can be put on monitors (i.e., VideoSink objects).

The VideoSource interface includes the SharedResource interface. A VideoSource is controlled by an Operations Center if the VideoSource is in maintenance mode, or if the VideoSource is a camera which has an active control session up.

5.3.1.1.39 VideoSwitch (Class)

The V1500Switch interface is implemented by a class representing any V1500 Video Switch in the CHART system. This interface provides access to configuration and status information for the switch, and provides connect and disconnect functions for making and breaking video connections.

5.3.1.1.40 VideoSwitchConfig (Class)

This represents the configuration information for a V1500 switch.

5.3.1.1.41 VideoTour (Class)

The Tour interface is implemented by a class which maintains the configuration and status of a single tour defined within the CHART II system.

5.3.1.1.42 VideoTourConfig (Class)

The TourConfig structure is used to hold and transmit configuration information about a given camera tour.

5.3.1.1.43 VideoTourEntry (Class)

The TourEntry structure is used to hold and transmit configuration information about a single entry in a camera tour.

5.3.1.1.44 VideoTourFactory (Class)

The TourManager interface is implemented by a class which tracks tours defined in the CHART II video system. It tracks the existence and configuration of tours and also tracks the status of all tours, whether they are active or not.

5.3.1.1.45 VideoTourState (Class)

The VideoTourState enumeration defines the values that can be used to indicate the status of a VideoTour.

5.3.1.1.46 VideoTourStatus (Class)

The TourStatus structure is used to hold and transmit status information about a given camera tour (e.g., what VideoSink objects the Tour is currently running on).

5.3.1.1.47 VideoTransmissionDevice (Class)

The VideoTransmissionDevice interface is implemented by objects representing devices which can be used for sending and receiving video. This interface provides CHART-standard methods for accessing status and configuration information. Specific interfaces supporting sending and receiving inherit from this abstract base interface.

5.3.1.2 VideoHighLevel-VideoSource (Class Diagram)

This diagram shows the VideoSource side of the VideoHighLevel diagram in more detail, adding Factories, Configuration and Status structures, exceptions, and other supporting interface elements. In general each of the major interface objects, VideoProvider, VideoSource, VideoCamera, and ControllableVideoCamera have a factory and configuration and status structures used to store and transmit configuration and status information to clients and interested server objects.



This enumeration identifies what action the camera is currently performing (if any).

This exception is thrown if an attempt to issue an immediate mode camera control command (such as pan, tilt, etc.) is issued while the camera is performing a long-running command (such as a `moveToPresetCommand` or a `setTitleCommand`). This indicates to the operator that the camera is momentarily busy, and the operator should try the action again in a few seconds, or when the camera image on the monitor shows that the long-running request has completed.

This exception is thrown if a request to control a camera is denied because the camera is already controlled, perhaps because a race condition where another operator has established control just before the request.

5.3.1.2.4 CameraNotControlledException (Class)

This is an exception thrown if an attempt to issue a camera control command is issued when the camera is not currently controlled by the requester. This is most likely to occur immediately after a control override, in cases where the client has not received or processed the override event yet.

5.3.1.2.5 CameraPreset (Class)

This structure stores information about a preset configured for a camera.

5.3.1.2.6 COHU3955Camera (Class)

The COHU3955Camera interface is implemented by objects representing COHU model 3055 video cameras. It extends the ControllableVideoCamera interface by adding methods unique to the COHU 3955 cameras (unique within the universe of camera types planned for implementation within CHART II).

5.3.1.2.7 COHU3955CameraStatus (Class)

The COHUCameraStatus structure is used to hold status information about COHUCamera objects at the COHUCamera level.

5.3.1.2.8 ControllableVideoCamera (Class)

The ControllableVideoCamera interface is implemented by objects representing controllable video cameras within the CHART II system. The ControllableVideoCamera interface represents a controllable video camera as opposed to an uncontrollable, immovable VideoCamera. Current plans call for classes to represent a COHU 3955 camera, Vicon SVFT camera, and NTCIP-compliant camera, and there are interfaces defined for each of these subtypes of ControllableVideoCamera. The ControllableVideoCamera interface includes all methods common to the two known types of video cameras currently in use by MDSHA, although it is likely to contain a superset of methods which would be implemented by the entire universe of all video cameras which could someday be used. This interface may have to be refined in the event that future brands or models of video cameras might be incorporated under CHART II, but it is an appropriate set of methods for the present day. Current plans call for classes to represent a COHU 3955 camera, Vicon SVFT camera, and NTCIP-compliant camera.

5.3.1.2.9 ControllableVideoCameraConfig (Class)

The ControllableVideoCameraConfig is used to hold and transmit configuration information about ControllableVideoCamera objects at the ControllableVideoCamera level.

5.3.1.2.10 ControllableVideoCameraStatus (Class)

The ControllableVideoCameraStatus is used to hold and transmit status information about ControllableVideoCameraStatus objects at the ControllableVideoCamera level.

5.3.1.2.11 ControllingInfo (Class)

The ControllingInfo structure contains information about the entity controlling (or requesting to control) a VideoCamera.

5.3.1.2.12 ControllingUserInfo (Class)

The ControllingUserInfo structure contains information about the monitor group and user of the entity controlling (or requesting to control) a VideoCamera.

5.3.1.2.13 DiagonallyMovable (Class)

The DiagonallyMovable interface is implemented by VideoCamera-enabled classes which can be moved diagonally in addition to standard orthogonal pan and tilt commands. A particular implementation may support 45-degree movements only, in which case the panDir and tiltDir parameters are +/- 1 to indicate direction only, or an implementation may support 360 degrees of motion, in which case, in addition to signs, the relative ratios of the parameters indicate the percent of movement proportionally in the pan/tilt directions. This interface is expected to be implemented beyond R2B2.

5.3.1.2.14 EnMasseSetResult (Class)

This structure will be used to communicate failures in setting a number of cameras to auto iris, auto focus, or auto color balance. It specifies results for one camera which failed.

5.3.1.2.15 EnMasseSetResultList (Class)

This structure will be used to communicate failures in setting a number of cameras to auto iris, auto focus, or auto color balance. It specifies results for all cameras which failed. (Cameras which succeeded are not included in this list.)

5.3.1.2.16 MonitorDisplayInfo (Class)

This structure holds details about each monitor on which the VideoProvider is currently being displayed.

5.3.1.2.17 NTCIPCamera (Class)

The NTCIPCamera interface is implemented by objects which support the NTCIP standard for CCTV cameras. As this is a future requirement for cameras not currently fielded by MDSHA, this interface is left to be defined at a later time.

5.3.1.2.18 PresetEnabled (Class)

The PresetEnabled interface is implemented by VideoCamera-enabled classes which can store and move to presets. The savePreset() method saves the current camera position as the preset position. This interface is expected to be implemented in R2B2.

5.3.1.2.19 PresetUndefinedException (Class)

This exception is thrown when an attempt is made to move to an undefined preset.

5.3.1.2.20 SharedResource (Class)

The SharedResource interface is implemented by any object that must always have an operations center responsible for the disposition of the resource while the resource is in use.

5.3.1.2.21 SharedResourceManager (Class)

The SharedResourceManager interface is implemented by classes that manage shared resources. Implementing classes must be able to provide a list of all shared resources under their management. Implementing classes must also be able to tell others if there are any resources under its management that are controlled by a given operations center.

5.3.1.2.22 TransferableSharedResource (Class)

The TransferrableSharedResource interface is implemented by any object that must always have an operations center responsible for the disposition of the resource while the resource is in use but may also be allowed to transfer control of that resource to another operations center.

5.3.1.2.23 ViconSVFTCamera (Class)

The ViconSVFTCamera interface is implemented by a class representing the Vicon Surveyor VFT model video camera. (As there are no other Vicon brand cameras used within CHART II, there is no base ViconCamera interface representing all Vicon-brand cameras. For one thing, there would be no known basis for allocating methods to the base interface and the VFT interface.)

5.3.1.2.24 ViconSVFTPgmCmd (Class)

The ViconSVFTPgmCmd enumeration defines the values that can be used in the programCommand() method of the ViconSVFTCamera interface.

5.3.1.2.25 VideoCamera (Class)

The VideoCamera interface is implemented by objects representing video cameras within the CHART II system. Classes implementing this interface (and nothing below this interface would be fixed (non-controllable) video cameras. The VideoCamera interface includes the GeoLocatable interface, to someday allow for advanced features such as automatic identification of cameras near traffic events, automatic pointing of cameras to traffic events, etc.

5.3.1.2.26 VideoCameraConfig (Class)

The VideoCameraConfig structure is used to hold configuration information about VideoCamera objects at the VideoCamera level. Further details about lower-level

VideoCamera subclasses are provided by subclasses of VideoCameraConfig. Device Location member has been modified for R3B3. Now it contains a detailed location information.

5.3.1.2.27 VideoCameraFactory (Class)

The GenericVideoCameraFactory interface is implemented by factory classes responsible for creating, maintaining, and controlling a collection of GenericVideoCamera objects.

5.3.1.2.28 VideoCameraStatus (Class)

The VideoCameraStatus structure is used to hold status information about VideoCamera objects at the VideoCamera level. Further details about lower-level VideoCamera subclasses are provided by subclasses of VideoCameraStatus.

5.3.1.2.29 VideoDisplayRevokedOrg (Class)

This structure is used to store information about an organization for which display of the associated camera has been revoked.

5.3.1.2.30 VideoProvider (Class)

The VideoProvider interface is a generic abstract interface including VideoSource objects (e.g. video cameras) and BridgeCircuit objects. Both VideoSource and BridgeCircuit objects provide video to a VideoCollector, but only VideoSource objects are true origins of video which a typical user would have direct interaction with. BridgeCircuit VideoProvider objects merely pass on video provided from elsewhere in a VideoRoute.

5.3.1.2.31 VideoProviderConfig (Class)

The VideoProviderConfig structure is used to hold configuration information about VideoProvider objects at the VideoProvider level. Further details about lower-level VideoProvider subclasses are provided by subclasses of VideoProviderConfig.

5.3.1.2.32 VideoProviderFactory (Class)

The VideoProviderFactory interface is implemented by factory classes responsible for creating and maintaining a collection of VideoProvider objects.

5.3.1.2.33 VideoProviderStatus (Class)

The VideoProviderStatus structure is used to hold status information about VideoProvider objects at the VideoProvider level. Further details about lower-level VideoProvider subclasses are provided by subclasses of VideoProviderStatus.

5.3.1.2.34 VideoSource (Class)

The VideoSource interface is implemented by objects which originate video signals, such as video cameras and image generators. Within the user interface, the VideoSource

interface represents all video sources which can be put on monitors (i.e., VideoSink objects).

The VideoSource interface includes the SharedResource interface. A VideoSource is controlled by an Operations Center if the VideoSource is in maintenance mode, or if the VideoSource is a camera which has an active control session up.

5.3.1.2.35 VideoSourceConfig (Class)

The VideoSourceConfig structure is used to hold configuration information about VideoSource objects at the VideoSource level. Further details about lower-level VideoSource subclasses are provided by subclasses of VideoSourceConfig.

5.3.1.2.36 VideoSourceFactory (Class)

The VideoSourceFactory interface is implemented by factory classes responsible for creating, maintaining, and controlling a collection of VideoSource objects.

5.3.1.2.37 VideoSourceStatus (Class)

The VideoSourceStatus structure is used to hold status information about VideoSource objects at the VideoSource level. Further details about lower-level VideoSource subclasses are provided by subclasses of VideoSourceStatus.

5.3.1.2.38 VideoSourceType (Class)

This enumeration identifies the various types of cameras which can exist in the system. The fixed type is for all non-controllable cameras.

5.3.1.3 CameraControlModule (Class Diagram)

This diagram shows the classes which comprise the CameraControlModule. The CameraControlModule is an installable module that serves the camera-type objects and factories to the rest of the CHART II system. This diagram shows how the implementation of these CORBA interfaces rely on other supporting classes to perform their functions. The CameraControlModule is responsible for serving all VideoSource objects including controllable cameras, fixed cameras, No Video Available sources, and potentially any other image generators, etc. The COHU3955CameraImpl and the viconSVFTCameraImpl are the primary classes operating in this module. These objects provide all access to the camera status and configuration. The CameraControlModule also includes factory implementations responsible for providing lists of cameras and other such objects to interested clients.

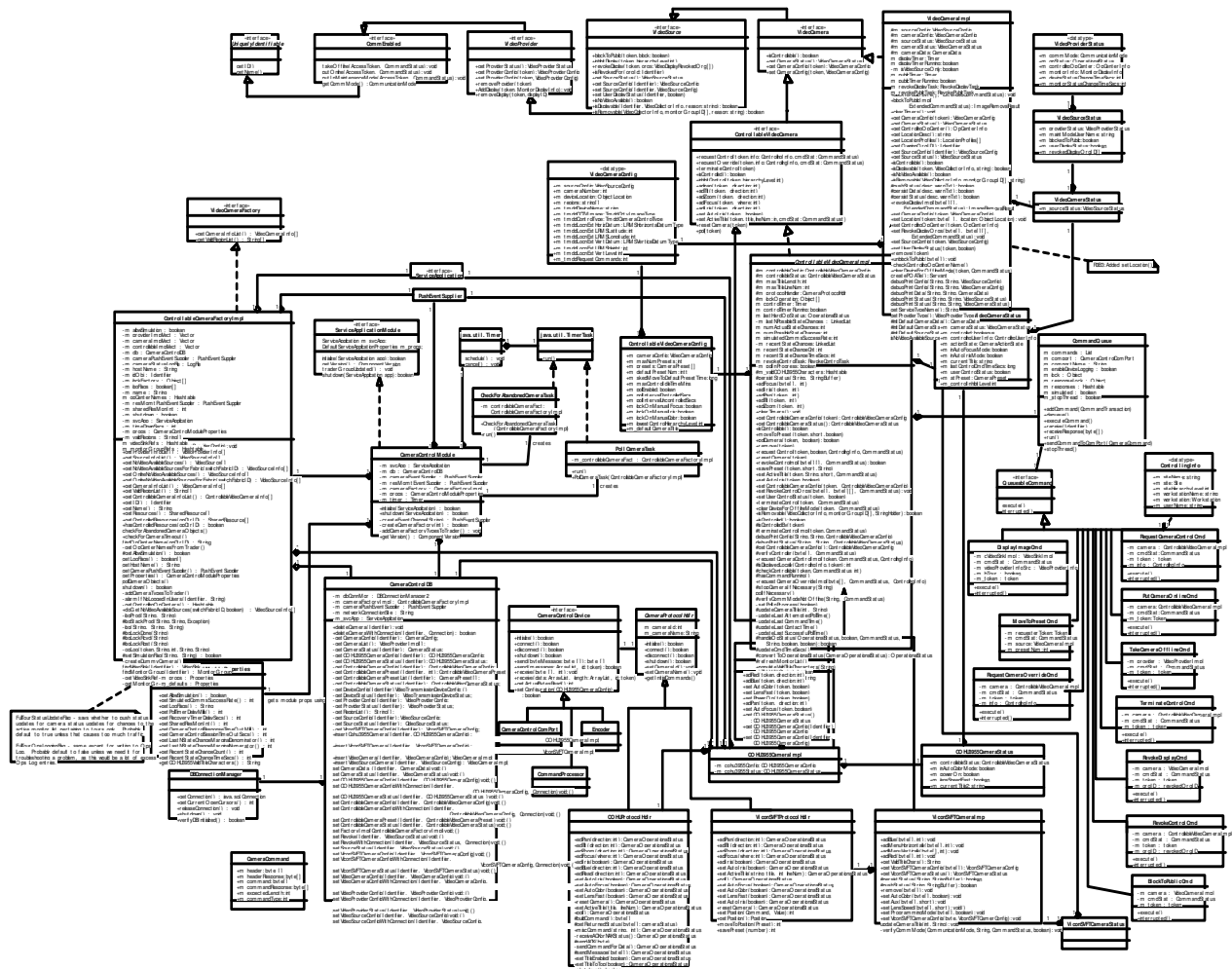


Figure 5-94 CameraControlModule (Class Diagram)

5.3.1.3.1 BlockToPublicCmd (Class)

This class represents the information needed to create a block camera to public command to be added on the CommandQueue.

5.3.1.3.2 CameraCommand (Class)

CameraCommand contains information about the commands sent to, and responses received from, the camera.

5.3.1.3.3 CameraControlComPort (Class)

The CameraControlComPort interface is implemented by a class representing a COM port with direct connection to the control port of a video camera. It is used to send video camera control commands and return responses to a camera control process.

5.3.1.3.4 CameraControlDB (Class)

The CameraControlDB class provides an interface between the Camera service and the database used to persist and depersist the Camera objects and their configuration and status in the database. It contains a collection of methods that perform database operations on tables pertinent to Camera Control. The class is constructed with a DBConnectionManager object, which manages database connections. Methods exist to insert and delete Camera objects from the database, and to get and set their configuration and status information.

5.3.1.3.5 CameraControlDevice (Class)

The CameraControlDevice interface is implemented by classes which provides communications for access to control functions for a video camera. This includes encoders, command processors, and direct COM ports.

5.3.1.3.6 CameraControlModule (Class)

The CameraControlModule class is the service module for the Camera devices and a Camera factory. It implements the ServiceApplicationModule interface. It creates and serves a single CameraFactoryImpl object, which in turn serves zero or more CameraImpl objects. It also creates CameraControlDB, CameraControlModuleProperties, and PushEventSupplier objects.

5.3.1.3.7 CameraControlModuleProperties (Class)

The CameraControlModuleProperties class is used to provide access to properties used by the Camera Control Module. This class wraps properties that are passed to it upon construction. It adds its own defaults and provides methods to extract properties specific to the Camera Control Module.

5.3.1.3.8 CameraProtocolHdlr (Class)

CameraProtocolHdlr classes provide implementations for all the camera commands. Each CameraImpl class will have a CameraProtocolHdlr instantiated when initialized. When a camera control command is sent to the CameraImpl, CameraProtocolHdlr will be called to translate the command to byte messages which the camera understands. Then those messages are sent by the CameraControlDevice to the camera. CameraProtocolHdlr is capable of using different CameraControlDevice which is created during the initialization.

5.3.1.3.9 CheckForAbandonedCameraTask (Class)

The CheckForAbandonedCameraTask is a timer task. When the timer fires, it checks to see if a camera control session has exceeded the timeout, or whether a camera is controlled by an Operations center with no one logged in.

5.3.1.3.10 COHU3955Camera (Class)

The COHUCamera interface is implemented by objects representing COHU-brand video cameras. The COHUCamera interface is extended by the COHUMPCCamera and

COHU3955Camera interfaces. The COHUCamera interface includes all methods which are common to the two COHU cameras used by CHART II, the COHU MPC camera and the COHU 3955 camera. (Note that this interface may well contain a superset of methods which would be implemented by the entire line of all models of COHU video cameras).

5.3.1.3.11 COHU3955CameraImpl (Class)

This class implements the COHU3955Camera interface, and inherits from the ControllableCameraImpl class. The COHU3955CameraImpl implements methods of COHU3955Camera, extending the controllable camera to include 3955-specific operations. This class will contain a configuration and status object as necessary to convey 3955-specific configuration and status information.

5.3.1.3.12 COHU3955CameraStatus (Class)

The CameraStatus class is an abstract value-type class which provides status information for a Camera. This status information is relatively dynamic: things like the communication mode, operational status, operation center information, status change time.

5.3.1.3.13 COHUProtocolHdlr (Class)

COHUProtocolHdlr is the base class for all COHU cameras. At present, this class contains implementations for common functions for COHU MPC and COHU 3955 cameras

5.3.1.3.14 CommandProcessor (Class)

The CommandProcessor class is used to send control commands to a camera.

5.3.1.3.15 CommandQueue (Class)

The CommandQueue class provides a queue for QueueableCommand objects. The CommandQueue has a thread that it uses to process each QueueableCommand in a first in first out order. As each command object is pulled off the queue by the CommandQueue's thread, the command object's execute method is called, at which time the command performs its intended task.

5.3.1.3.16 CommEnabled (Class)

The CommEnabled interface is implemented by objects that can be taken offline, put online, or put in maintenance mode through a standard interface. These states typically apply only to field devices. When a device is taken offline, it is no longer available for use through the system and automated polling (if any) is halted. When put online, a device is again available for use by TrafficEvents within the system and automated polling is enabled (if applicable). When put in maintenance mode a device is offline (i.e., cannot be used by TrafficEvents), and maintenance commands appropriate for the particular type of device are allowed to help in troubleshooting.

5.3.1.3.17 ControllableCameraFactoryImpl (Class)

The CameraFactoryImpl class provides an implementation of the CameraFactory interface (and its CameraFactory and SharedResourceManager interfaces) as specified in the IDL. The CameraFactoryImpl maintains a list of CameraImpl objects and is responsible for publishing Camera objects in the Trader on startup and as new camera objects are created. Whenever a Camera is created or removed, that information is persisted to the database. This class is also responsible for performing the checks requested by the timer tasks: to poll the Camera devices, to look for Camera devices with timeout exceeded, to look for Camera devices with no one logged in at the controlling operations center, and to initiate recovery processing as needed

5.3.1.3.18 ControllableVideoCamera (Class)

The ControllableVideoCamera interface is implemented by objects representing controllable video cameras within the CHART II system. The ControllableVideoCamera interface represents a controllable video camera as opposed to the uncontrollable, immovable VideoCamera. Current plans call for classes to represent a COHU MPC camera, COHU 3955 camera, Vicon SVFT camera, and NTCIP-compliant camera, and there are interfaces defined for each of these subtypes of ControllableVideoCamera. The ControllableVideoCamera interface includes all methods common to the three known types of video cameras currently in use by MDSHA, although it is likely to contain a superset of methods which would be implemented by the entire universe of all video cameras which could someday be used. This interface may have to be refined in the event that future brands or models of video cameras might be incorporated under CHART II, but it is an appropriate set of methods for the present day. Current plans call for classes to represent a COHU MPC camera, COHU 3955 camera, Vicon SVFT camera, and NTCIP-compliant camera.

5.3.1.3.19 ControllableVideoCameraConfig (Class)

The ControllableVideoCameraConfig is used to hold and transmit configuration information about ControllableVideoCamera objects at the ControllableVideoCamera level.

5.3.1.3.20 ControllableVideoCameraImpl (Class)

The ControllableCameraImpl class provides an implementation of the ControllableVideoCamera interface and is derived from the CameraImpl class implementing the VideoCamera interface.

This class contains a CommandQueue object that is used to sequentially execute long running operations related to camera control in a thread separate from the CORBA request threads, thus allowing quick initial responses.

Also contained in this class are ControllableVideoCameraConfig and ControllableVideoCameraStatus objects (used to store the configuration and status of the camera), and a VideoCameraData object (used to store internal status information which is persisted but not pushed out to clients).

The ControllableCameraImpl contains *Impl methods that map to methods specified in the IDL, including requests to request control of the camera, terminate control of the camera, override control of the camera, and to send pan/tilt/zoom (PTZ) commands to the camera. Some of these requests are long running, so each request is stored in a specific subclass of QueueableCommand and added to the CommandQueue. The queueable command objects simply call the appropriate ControllableCameraImpl method as the command is executed by the CommandQueue in its thread of execution. PTZ commands are not considered long running and are not placed on the command queue.

The ControllableCameraImpl also contains methods called by the CameraFactory to support the timer tasks of the Camera Service: to poll the Camera, to look for Camera devices with communications timeout exceeded.

5.3.1.3.21 ControllableVideoCameraStatus (Class)

The ControllableVideoCameraStatus is used to hold and transmit status information about ControllableVideoCameraStatus objects at the ControllableVideoCamera level.

5.3.1.3.22 ControllingInfo (Class)

The ControllingInfo structure contains information about the entity controlling (or requesting to control) a VideoCamera.

5.3.1.3.23 DBConnectionManager (Class)

This class implements a database connection manager that manages a pool of database connections. Any CHART II system thread requiring database access gets a database connection from the pool of connections maintained by this manager class. The connections are maintained in two separate lists namely, inUseList and freeList. The inUseList contains connections that have already been assigned to a thread. The freeList contains unassigned connections. This class assumes that an appropriate JDBC driver has been loaded either by using the "jdbc.drivers" system property or by loading it explicitly. The class has a monitor thread that is started by the constructor. This connection monitor thread periodically checks the inuseList to see if there are connections that are owned by dead threads and move such connections to the freeList. The connection monitor thread is started only if a non-zero value is specified for the monitoring time interval in the constructor.

5.3.1.3.24 DisplayImageCmd (Class)

This class represents the information needed to create a display image command to be added on the CommandQueue.

5.3.1.3.25 Encoder (Class)

The Encoder interface is implemented by classes representing any type of video encoder. The Encoder interface includes both the Codec and the VideoSendingDevice interfaces, which means in addition to providing forwarding of video, it also is used to send video camera control commands and return responses to a camera control process.

5.3.1.3.26 java.util.Timer (Class)

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

5.3.1.3.27 java.util.TimerTask (Class)

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

5.3.1.3.28 MoveToPresetCmd (Class)

This class represents the information needed to create a move to preset command to be added on the CommandQueue.

5.3.1.3.29 PollCameraTask (Class)

The PollCameraTask is a timer task. When the timer fires it polls a camera by sending a poll command to the camera.

5.3.1.3.30 PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.3.1.3.31 PutCameraOnlineCmd (Class)

This class represents the information needed to request a put camera online command to be added on the CommandQueue.

5.3.1.3.32 QueueableCommand (Class)

A QueueableCommand is an interface used to represent a command that can be placed on a CommandQueue for asynchronous execution. Derived classes implement the execute method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a CommandQueue. The CommandQueue driver calls the execute method to execute a command in the queue and a call to the interrupted method is made when a CommandQueue is shut down.

5.3.1.3.33 RequestCameraControlCmd (Class)

This class represents the information needed to request a camera control command to be added on the CommandQueue.

5.3.1.3.34 RequestCameraOverrideCmd (Class)

This class represents the information needed to request a camera control override command to be added on the CommandQueue.

5.3.1.3.35 RevokeControlCmd (Class)

This class represents the information needed to create a revoke camera control command to be added on the CommandQueue.

5.3.1.3.36 RevokeDisplayCmd (Class)

This class represents the information needed to create a revoke camera display command to be added on the CommandQueue.

5.3.1.3.37 ServiceApplication (Class)

This interface is implemented by objects that can provide the basic services needed by a ChartII service application. These services include providing access to basic CORBA objects that are needed by service applications, such as the ORB, POA, Trader, and Event Service.

5.3.1.3.38 ServiceApplicationModule (Class)

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

5.3.1.3.39 TakeCameraOfflineCmd (Class)

This class represents the information needed to request a take camera offline command to be added on the CommandQueue.

5.3.1.3.40 TerminateControlCmd (Class)

This class represents the information needed to request a terminate camera control command to be added on the CommandQueue.

5.3.1.3.41 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.3.1.3.42 ViconSVFTCameraImpl (Class)

This class implements the ViconSVFTCamera interface, and inherits from the ControllableCameraImpl class. The ViconSurveyorVFTCameraImpl implements methods of ViconSVFTCamera, extending the controllable camera to include Vicon SVFT-specific operations. This class will contain a configuration and status object as necessary to convey Vicon SVFT-specific configuration and status information.

5.3.1.3.43 ViconSVFTCameraStatus (Class)

The ViconSVFTCameraStatus class is used to hold camera status information at the ViconSVFTCamera level. Only ViconSVFTCamera specific information is stored.

5.3.1.3.44 ViconSVFTProtocolHdlr (Class)

This class contains an implementation for Vicon SVFT camera control commands. It translates every camera command (pan, tilt, zoom...) into bytes that a Vicon SVFT camera understands. Then, it uses a CameraControlDevice to send the byte codes to the camera and evaluate responses from the camera.

5.3.1.3.45 VideoCamera (Class)

The VideoCamera interface is implemented by objects representing controllable video cameras within the CHART II system. The VideoCamera interface represents a controllable video camera as opposed to the uncontrollable, immovable FixedVideoCamera, the other type of GenericVideoCamera. (The VideoCamera class could have been called the ControllableVideoCamera interface, but since the CHART II video system exists primarily to control controllable video cameras, the camera hierarchy has been arranged to avoid the longish name ControllableVideoCamera.) Current plans call for classes to represent a COHU MPC camera, COHU 3955 camera, Vicon SVFT camera, and NTCIP-compliant camera, and there are interfaces defined for each of these subtypes of VideoCamera. The VideoCamera interface includes the GeoLocatable interface, to someday allow for advanced features such as automatic identification of cameras near traffic events, automatic pointing of cameras to traffic events, etc.

The VideoCamera interface includes all methods common to the three known types of video cameras currently in use by MDSHA, although it is likely to contain a superset of methods which would be implemented by the entire universe of all video cameras which could someday be used. This interface may have to be refined in the event that future brands or models of video cameras might be incorporated under CHART II, but it is an appropriate set of methods for the present day.

5.3.1.3.46 VideoCameraConfig (Class)

The VideoCameraConfig structure is used to hold configuration information about VideoCamera objects at the VideoCamera level. Further details about lower-level VideoCamera subclasses are provided by subclasses of VideoCameraConfig.

5.3.1.3.47 VideoCameraFactory (Class)

The VideoCameraFactory interface is implemented by factory classes responsible for creating, maintaining, and controlling a collection of VideoCamera objects.

5.3.1.3.48 VideoCameraImpl (Class)

The CameraImpl class provides an implementation of the VideoCamera interface, and by extension the VideoSource, SharedResource, CommEnabled, GeoLocatable, and UniquelyIdentifiable interfaces, as specified by the IDL.

This class contains a CommandQueue object that is used to sequentially execute long running operations in a thread separate from the CORBA request threads, thus allowing quick initial responses.

Also contained in this class are VideoCameraConfig and VideoCameraStatus objects (used to store the configuration and status of the camera), and a VideoCameraData object (used to store internal status information which is persisted but not pushed out to clients).

The CameraImpl contains *Impl methods that map to methods specified in the IDL, including requests to display the camera video on a monitor, remove the camera video from a monitor, put the camera online, put the camera offline, put the camera in maintenance mode (future), or to change (set) the configuration of the camera (future). Some of these requests require (or potentially require) field communications to the device, so each request is stored in a specific subclass of QueueableCommand and added to the CommandQueue. The queueable command objects simply call the appropriate CameraImpl method as the command is executed by the CommandQueue in its thread of execution.

The CameraImpl also contains methods called by the CameraFactory to support the timer tasks of the Camera Service: to look for Cameras with no one logged in at the controlling operations center, and to initiate recovery processing if needed (future).

5.3.1.3.49 VideoCameraStatus (Class)

The VideoCameraStatus structure is used to hold status information about VideoCamera objects at the VideoCamera level. Further details about lower-level VideoCamera subclasses are provided by subclasses of VideoCameraStatus.

5.3.1.3.50 VideoProvider (Class)

The VideoProvider interface is a generic abstract interface including VideoSource objects (e.g. video cameras) and BridgeCircuit objects. Both VideoSource and BridgeCircuit objects provide video to a VideoCollector, but only VideoSource objects are true origins of video which a typical user would have direct interaction with. BridgeCircuit VideoProvider objects merely pass on video provided from elsewhere in a VideoRoute.

5.3.1.3.51 VideoProviderStatus (Class)

The VideoProviderStatus structure is used to hold and transmit status information about

VideoProvider objects at the VideoProvider level. Further details about lower-level VideoProvider subclasses are provided by subclasses of VideoProviderStatus.

5.3.1.3.52 VideoSource (Class)

The VideoSource interface is implemented by objects which originate video signals, such as video cameras and image generators. Within the user interface, the VideoSource interface represents all video sources which can be put on monitors (i.e., VideoSink objects).

The VideoSource interface includes the SharedResource interface. A VideoSource is controlled by an Operations Center if the VideoSource is in maintenance mode, or if the VideoSource is a camera which has an active control session up.

5.3.1.3.53 VideoSourceStatus (Class)

The VideoSourceStatus structure is used to hold and transmit status information about VideoSource objects at the VideoSource level. Further details about lower-level VideoSource subclasses are provided by subclasses of VideoSourceStatus.

5.3.2 Sequence Diagrams

5.3.2.1 CameraControlModule:SetCameraConfiguration (Sequence Diagram)

This sequence diagram shows the implementation of the setConfiguration interface of the CameraImpl class (which represents VideoProviderImpl, VideoSourceImpl, VideoCameraImpl etc.). First a check is performed to verify that the operator has sufficient privileges to update a camera. Next a check is made to see that the camera is offline. Only offline cameras may have their configurations updated. If the camera is offline, the new configuration is validated. Next the new configuration is written to the database. Finally, the camera is apprised of its new configuration.

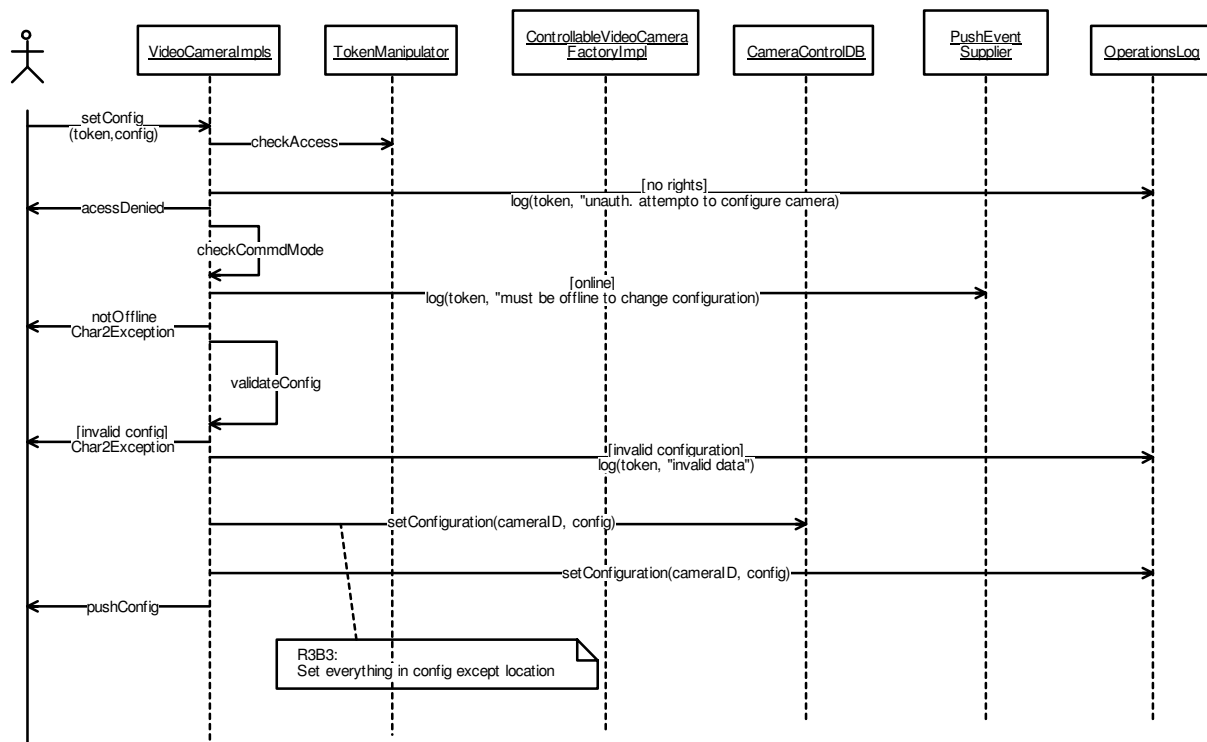


Figure 5-95 CameraControlModule:SetCameraConfiguration (Sequence Diagram)

5.4.1.1.1 AlertFactoryWrapper (Class)

This singleton class provides a wrapper for the Alert Factory that provides automatic location of an Alert Factory and automatic re-discovery should the Alert Factory reference return an error. This class also allows for built-in fault tolerance by automatically failing over to a "working" Alert Factory without the user of this class being aware that this is being done. In addition, this class defers the discovery of the Alert Factory until its first use, thus eliminating a start-up dependency for modules that use the Alert Factory.

This class delegates all of its method calls to the system AlertFactory using its currently known good reference to an AlertFactory. If the current reference returns a CORBA failure in the delegated call, this class automatically switches to another reference. When there are no good references (as is true the first time the object is used), this class issues a trader query to (re)discover the published Alert Factory objects in the system. During a method call, the trader will be queried at most one time and under normal circumstances, not at all.

5.4.1.1.2 ArbitrationQueue (Class)

An ArbitrationQueue is a queue that arbitrates the usage of a device. The evaluation of the queue determines which message(s) should be on the device, based upon the priority of the queue entries. When entries are added to the queue, they are assigned a priority level based on the type of traffic event with which they are associated, and also upon the current contents of the queue. The priority of the queue entries can be modified after they are added to the queue. The queue is evaluated when the device is online and queue entries are added or removed, when an entry's priority is modified, or when the device is put online.

5.4.1.1.3 ArbQueueEntry (Class)

This class is used for an entry on the arbitration queue, for a single message, and for a single traffic event. (It is possible, in the case of HARNotifierArbQueueEntry objects, that certain ArbQueueEntries can be on behalf of multiple TrafficEvents. In such cases, one TrafficEvent among all those involved is picked to be the responsible TrafficEvent stored in m_indicator, the ArbQueueEntryIndicator for the entry.)

5.4.1.1.4 Chart2DMS (Class)

The Chart2DMS class extends the DMS interface and defines a more detailed interface to be used in manipulating the CHART-specific DMS objects within CHART. It provides an interface for traffic events to provide input as to what each traffic event desires to be on the sign via the ArbitrationQueue interface. Through the HARMessageNotifier interface, a HAR can use a DMS to notify travelers to tune in to a radio station to hear a traffic message. CHART business rules include concepts such as shared resources, arbitration queues, and linking device usage to traffic events. These concepts go beyond industry-standard DMS control. This includes an ability to enable and disable CHART traveler information messages, which were added in R3B3.

5.4.1.1.5 Chart2DMSConfiguration (Class)

The Chart2DMSConfiguration class is an abstract class which extends the DMSConfiguration class to provide configuration information specific to Chart II processing. Such information includes how to contact the sign under Chart II software control, the default SHAZAM message for using the sign as a HAR Notifier, and the owning organization. Such data extends beyond what would be industry-standard configuration information for a DMS. Parameters to support TCP/IP communications, notifications and more alerts, and traveler information messages were added for R3B3.

5.4.1.1.6 Chart2DMSData (Class)

This class is used to store data associated with a DMS such as last contact time etc

5.4.1.1.7 Chart2DMSFactory (Class)

The Chart2DMSFactory interface extends the DMSFactory interface to provide additional Chart II specific capability. This factory creates Chart2DMS objects (extensions of DMS objects). It implements the SharedResourceManager capability to control DMS objects as shared resources.

5.4.1.1.8 Chart2DMSFactoryImpl (Class)

The Chart2DMSFactoryImpl class provides an implementation of the Chart2DMSFactory interface (and its DMSFactory and SharedResourceManager interfaces) as specified in the IDL. The Chart2DMSFactoryImpl maintains a list of Chart2DMSImpl objects and is responsible for publishing DMS objects in the Trader on startup and as new DMS objects are created. Whenever a DMS is created or removed, that information is persisted to the database. This class is also responsible for performing the checks requested by the timer tasks: to poll the DMS devices, to look for DMS devices with timeout exceeded, to look for DMS devices with no one logged in at the controlling operations center, and to initiate recovery processing as needed.

5.4.1.1.9 Chart2DMSImpl (Class)

The Chart2DMSImpl class provides an implementation of the Chart2DMS interface, and by extension the DMS, SharedResource, HARMessageNotifier, CommEnabled, GeoLocatable, ArbitrationQueue and UniquelyIdentifiable interfaces, as specified by the IDL.

This class contains a CommandQueue object that is used to sequentially execute long running operations (field communications to the device) in a thread separate from the CORBA request threads, thus allowing quick initial responses. The Chart2DMSImpl also contains a MessageQueue, which is used by the ArbitrationQueue interface methods to handle requests from TrafficEvents to display or remove messages from the signs in online mode. When the Chart2DMSImpl evaluates its messages in the MessageQueue, it combines the highest priority messages into a single message which is placed into an appropriate QueueableCommand object (subclass of QueueableCommand) and added to the CommandQueue.

Also contained in this class are Chart2DMSTConfiguration and Chart2DMSTStatus objects (used to store the configuration and status of the sign), and a Chart2DMSTData object (used to store internal status information which is persisted but not pushed out to clients), a list of ArbQueueEntry objects from the MessageQueue that are currently active on the sign, and a copy of the last QueueableCommand to put a message on the sign.

The Chart2DMSTImpl contains *Impl methods that map to methods specified in the IDL, including requests to put a message on the sign or remove a message (in maintenance mode only), put the sign online, put the sign offline, put the sign in maintenance mode, or to change (set) the configuration of the sign. All of these requests require (or potentially require) field communications to the device, so each request is stored in a specific subclass of QueueableCommand and added to the CommandQueue. The queueable command objects simply call the appropriate Chart2DMSTImpl method as the command is executed by the CommandQueue in its thread of execution.

The Chart2DMSTImpl also contains methods called by the Chart2DMSTFactory to support the timer tasks of the DMS Service: to poll the DMS devices, to look for DMS devices with communications timeout exceeded, to look for maintenance mode DMS devices with no one logged in at the controlling operations center, and to initiate recovery processing if needed.

5.4.1.1.10 Chart2DMSTStatus (Class)

The Chart2DMSTStatus class is an abstract class which extends the DMSTStatus class to provide status information specific to CHART processing, such as information on the controlling operations center for the sign. This data extends beyond what would be industry-standard status information for a DMS. Status information for traveler information messages was added in R3B3.

5.4.1.1.11 CheckCommLossTask (Class)

The CheckCommLossTask class is responsible for determining when communications to a DMS device have been down long enough to decide that the sign is or should be blank or considered to be blank. The anticipated time interval for making such a determination is on the order of ten minutes (however, this task is called much more frequently than that, so that the timeout can be detected soon after it has expired). This class implements the java.util.TimerTask interface, and as such it contains one method, run(), which is invoked by Java timer object on a regularly scheduled basis. This class contains a reference to the Chart2DMSTFactoryImpl, which is called upon to actually check the DMS objects each time this task is called.

5.4.1.1.12 CheckForAbandonedDMSTask (Class)

The CheckForAbandonedDMSTask class is responsible for detecting any DMS device in maintenance mode with a message on it which has no one logged in at the controlling operations center. This would only occur as a result of an anomaly, such as a reboot of a user's machine, because during a normal Chart II logout attempt, the logout is prohibited by

Chart II system if the user is the last user on his/her operations center and that operations center is controlling a sign. However, because anomalies happen, this task runs periodically to look for abandoned DMS devices. This class implements the `java.util.TimerTask` interface, and as such it contains one method, `run()`, which is invoked by Java timer object on a regularly scheduled basis. This class contains a reference to the `Chart2DMSFactoryImpl`, which is called upon to actually check the DMS objects and controlling operations centers of each DMS every time this task is called.

5.4.1.1.13 CommandQueue (Class)

The `CommandQueue` class provides a queue for `QueueableCommand` objects. The `CommandQueue` has a thread that it uses to process each `QueueableCommand` in a first in first out order. As each command object is pulled off the queue by the `CommandQueue`'s thread, the command object's `execute` method is called, at which time the command performs its intended task.

5.4.1.1.14 CommEnabled (Class)

The `CommEnabled` interface is implemented by objects that can be taken offline, put online, or put in maintenance mode through a standard interface. These states typically apply only to field devices. When a device is taken offline, it is no longer available for use through the system and automated polling (if any) is halted. When put online, a device is again available for use by `TrafficEvents` within the system and automated polling is enabled (if applicable). When put in maintenance mode a device is offline (i.e., cannot be used by `TrafficEvents`), and maintenance commands appropriate for the particular type of device are allowed to help in troubleshooting.

5.4.1.1.15 DBConnectionManager (Class)

This class implements a database connection manager that manages a pool of database connections. Any CHART II system thread requiring database access gets a database connection from the pool of connections maintained by this manager class. The connections are maintained in two separate lists namely, `inUseList` and `freeList`. The `inUseList` contains connections that have already been assigned to a thread. The `freeList` contains unassigned connections. This class assumes that an appropriate JDBC driver has been loaded either by using the `"jdbc.drivers"` system property or by loading it explicitly. The class has a monitor thread that is started by the constructor. This connection monitor thread periodically checks the `inuseList` to see if there are connections that are owned by dead threads and move such connections to the `freeList`. The connection monitor thread is started only if a non-zero value is specified for the monitoring time interval in the constructor.

5.4.1.1.16 DictionaryWrapper (Class)

This singleton class provides a wrapper for the system dictionary that provides automatic location of the dictionary and automatic re-discovery should the dictionary reference return an error. This class also allows for built-in fault tolerance by automatically failing over to a "working" dictionary without the user of this class being aware that this being done. In

addition, this class defers the discovery of the Dictionary until its first use, thus eliminating a start-up dependency for modules that use the dictionary.

This class delegates all of its method calls to the system dictionary using its currently known good reference to the system dictionary. If the current reference returns a CORBA failure in the delegated call, this class automatically switches to another reference. When there are no good references (as is true the first time the object is used), this class issues a trader query to (re)discover the published Dictionary objects in the system. During a method call, the trader will be queried at most one time and under normal circumstances (other than the first use) the trader will not be queried at all.

5.4.1.1.17 DiscoveryManager (Class)

This SystemContextProvider interface defines some of the functionality required by a class which provides discovery services for CHART services. It is used by both the CHART GUI and the CHART backend services. A class which implements this interface must provide "get" accessor methods for the system profile properties, the data model, and the main processing queue for a service, for instance. It also provides access to the root deployment path and dynamic image path, which is used only by the CHART GUI. For the CHART GUI, this interface is known to be implemented by the MainServlet; for the backend CHART services, this interface is known to be implemented by the Discovery Manager.

5.4.1.1.18 DMS (Class)

The DMS class defines an interface to be used in manipulating Dynamic Message Sign (DMS) objects within Chart II. It specifies methods for setting messages and clearing messages from a sign (in maintenance mode), polling a sign, changing the configuration of a sign, and resetting a sign. (Setting messages on a sign in online mode are not accomplished by manipulating a DMS directly; that is accomplished by manipulating traffic events, which use an ArbitrationQueue interface or by manipulating HARs, which use a HARMessageNotifier interface. This activity involves the DMS extension, Chart2DMS, which defines interactions with signs under Chart II business rules.)

5.4.1.1.19 DMSArbitrationQueue (Class)

This class provides the implementation of an arbitration queue tailored for DMS devices.

5.4.1.1.20 DMSControlDB (Class)

The DMSControlDB class provides an interface between the DMS service and the database used to persist the DMS objects and their configuration and status in the database. It contains a collection of methods that perform database operations on tables pertinent to DMS Control. The class is constructed with a DBConnectionManager object, which manages database connections. Methods exist to insert and delete DMS objects from the database, and to get and set their configuration and status information. All information about a sign is persisted, including its current displayed message, communications status, and time of last contact, so that a momentary glitch or restart of the software will not

interrupt messages on signs.

5.4.1.1.21 DMSControlModule (Class)

The DMSControlModule class is the service module for the DMS devices and a DMS factory. It implements the ServiceApplicationModule interface. It creates and serves a single DMSFactoryImpl object, which in turn serves zero or more Chart2DMSImpl objects. It also creates DMSControlDB, DictionaryWrapper, DMSControlModuleProperties, and PushEventSupplier and NotificationChannel objects.

5.4.1.1.22 DMSControlModuleProperties (Class)

The DMSControlModuleProperties class is used to provide access to properties used by the DMS Control Module. This class wraps properties that are passed to it upon construction. It adds its own defaults and provides methods to extract properties specific to the DMS Control Module.

5.4.1.1.23 DMSFactory (Class)

The DMSFactory class specifies the interface to be used to create DMS objects within the Chart II system. It also provides a method to get a list of DMS devices currently in the system.

5.4.1.1.24 DMSImpl (Class)

This abstract class implements the DMS interface. It provides methods to configure and control DMS objects in CHART. This class was added in R3B3, when the concept of external DMSs was added to CHART. The DMSImpl class is the base class for both Chart2DMSImpl and ExternalDMSImpl, and provides methods shared by these two classes.

5.4.1.1.25 DMSTravInfoMsg (Class)

This class holds information necessary to put traveler information messages (containing travel times and/or toll rates) on DMSs. Each TravelerInfoMsg contains the ID for the template, and the IDs of the routes to use, as configured for its specific DMS. Each TravelerInfoMsg can be enabled or disabled. The DMSControlModule will ensure that a maximum of one TravelerInfoMsg is enabled at a time.

5.4.1.1.26 DMSTravInfoMsgDataSupplier (Class)

This interface provides data for travel routes used in a DMSTravInfoMsg. It will be used to substitute the template tags with route-specific data, in order to format the template and produce MULTI. This is needed in the GUI for true display, and is needed in the server for formatting messages to send to a DMS. The routeNum parameter corresponds to route numbers contained in the template data tags, and it is a 1-based index. These methods will throw an exception if the requested data is not available.

5.4.1.1.27 DMSTravInfoMsgHandler (Class)

This class implements DMSTravInfoMsgDataSupplier interface. Class will provide data for travel routes used in a DMSTravInfoMsg. It will be used to substitute the template tags with route-specific data, in order to format the template and produce MULTI. This is needed in the GUI for true display, and is needed in the server for formatting messages to send to a DMS. The routeNum parameter corresponds to route numbers contained in the template data tags, and it is a 1-based index. These methods will throw an exception if the requested data is not available.

5.4.1.1.28 DMSTravInfoMsgTemplateModel (Class)

This class contains functionality for formatting and modelling DMS message templates. During initialization a model of pages, rows, and elements (including the template tags) is constructed. MULTI fragments (the MULTI outside of the template tags) are stored so that they can be carried to the formatted MULTI. The tags can also be queried from the model, which can be used to figure out what data will be required for each route by the template.

5.4.1.1.29 ExternalDMS (Class)

The ExternalDMS class extends the DMS interface and defines a more detailed interface to be used in manipulating the External DMS objects within CHART.

5.4.1.1.30 ExternalDMSFactory (Class)

The ExternalDMSFactory interface extends the DMSFactory interface.. This factory creates ExternalDMS objects (extensions of DMS objects).

5.4.1.1.31 ExternalDMSFactoryImpl (Class)

This class implements the ExternalDMSFactory interface. It provides the interface to create, remove and list ExternalDMS objects in CHART mirroring the data from external agencies.

5.4.1.1.32 ExternalDMSImpl (Class)

This class implements the ExternalDMS interface. It provides the interface to ExternalDMS objects in CHART.

5.4.1.1.33 FP9500DMS (Class)

The FP9500DMS class extends the Chart2DMS interface and defines a more detailed interface to be used in manipulating FP9500 models of DMS signs. It is exemplary of potentially a whole suite of subclasses specific to a specific brand and model of sign for manufacturer-specific DMS control. For instance, the FP9500DMS has a performPixelTest method, which knows how to invoke and interpret a pixel test as supported by the FP9500 model DMS.

5.4.1.1.34 FP9500DMSImpl (Class)

The FP9500DMSImpl class provides a specific implementation to implement the FP9500DMS interface, providing any specific functionality unique to this brand and model of sign. This class is exemplary of a whole suite of implementation classes which may be created, on a case-by-case basis, to support specific capabilities of specific brands and models of signs.

5.4.1.1.35 FP9500DMSStatus (Class)

The FP9500DMSStatus class provides additional storage for status information unique to the FP9500 model of sign. It is exemplary of potentially a whole suite of Chart2DMSSStatus subclasses specific to a specific brand and model of sign.

5.4.1.1.36 GeoLocatable (Class)

This interface is implemented by objects that can provide location information to their users.

5.4.1.1.37 HARMessageNotifier (Class)

The HARMessageNotifier class specifies an interface to be implemented by devices that can be used to notify the traveler to tune in to a radio station to hear a traffic message being broadcast by a HAR. A HARMessageNotifier is directional and allows users of the device to better determine if activation of the device is warranted for the message being broadcast by the HAR. This interface can be implemented by SHAZAM devices and by DMS devices which are allowed to provide a SHAZAM-like message.

5.4.1.1.38 java.util.Properties (Class)

The Properties class represents a persistent set of properties. The Properties can be saved to a stream or loaded from a stream. Each key and its corresponding value in the property list is a string. A property list can contain another property list as its "defaults"; this second property list is searched if the property key is not found in the original property list.

5.4.1.1.39 java.util.Timer (Class)

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

5.4.1.1.40 java.util.TimerTask (Class)

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

5.4.1.1.41 PollDMSTask (Class)

The PollDMSTask class is responsible for polling all the DMS devices. This class implements the java.util.TimerTask interface, and as such it contains one method, run(),

which is invoked by Java timer object on a regularly scheduled basis. This class contains a reference to the Chart2DMSFactoryImpl, which is called upon to request each DMS to poll itself (its poll interval has expired) each time this task is called.

5.4.1.1.42 PortLocator (Class)

The PortLocator is a utility class that helps one to connect to the port used by the device. The actual implementation of the operations is done by the derived classes depending on what protocol is used for communication.

5.4.1.1.43 PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.4.1.1.44 QueueableCommand (Class)

A QueueableCommand is an interface used to represent a command that can be placed on a CommandQueue for asynchronous execution. Derived classes implement the execute method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a CommandQueue. The CommandQueue driver calls the execute method to execute a command in the queue and a call to the interrupted method is made when a CommandQueue is shut down.

5.4.1.1.45 RecoveryTimerTask (Class)

This Timer Task runs on a regular basis (on the order of every 15-30 seconds) during the life of the process. During normal operations, this task's sole purpose is to write a timestamp to a file each time it is called. This timestamp file serves to provide, to an approximation as accurate as its frequency of invocation, when the DMSService last went down, an essential piece of information for recovery during DMSService startup. When the DMSService has recently started up, this Task, in addition to maintaining an up-to-date timestamp in the timestamp file, also calls a method in the Factory (checkDMSRecovery) which requests all DMS objects to check and see if their recovery period has expired. (The recovery period is defined to be their poll interval times a system-wide multiplier (expected to be 2), or, if the DMS has no poll interval, a system-wide constant (on the order of 10-15 minutes.) Each DMS, therefore terminates its recovery period independently of the others.

(When all DMSes have terminated their recovery period, checkDMSRecovery is no longer called.)

When each DMS checks its own recovery time, if it finds that it has just now exceeded the recovery period, it calls its MessageQueue to take one last try at resolving traffic events on its queue, then the DMS makes final a determination as to what message (or blank) belongs on the sign, and it requests the DMS to set the sign appropriately.

5.4.1.1.46 ServiceApplication (Class)

This interface is implemented by objects that can provide the basic services needed by a ChartII service application. These services include providing access to basic CORBA objects that are needed by service applications, such as the ORB, POA, Trader, and Event Service.

5.4.1.1.47 ServiceApplicationModule (Class)

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

5.4.1.1.48 SetDMSMessageFromQueueCmd (Class)

The SetDMSMessageFromQueueCmd class is a QueueableCommand subclass which contains data necessary to send a request to a Chart2DMSImpl to put a message on the sign during normal operations (online mode). It is created by the Chart2DMSImpl during successful processing of its setMessageFromQueue and evaluateQueue methods. When the CommandQueue invokes the execute method of this class, it merely calls the setDMSMessageFromQueueImpl method of the appropriate Chart2DMSImpl object with the data stored within this class.

5.4.1.1.49 SharedResource (Class)

The SharedResource interface is implemented by any object that may have an operations center responsible for the disposition of the resource while the resource is in use.

5.4.1.1.50 SharedResourceManager (Class)

The SharedResourceManager interface is implemented by classes that manage shared resources. Implementing classes must be able to provide a list of all shared resources under their management. Implementing classes must also be able to tell others if there are any resources under its management that are controlled by a given operations center. The shared resource manager is also responsible for periodically monitoring its shared resources to detect if the operations center controlling a resource doesn't have at least one user logged into the system. When this condition is detected, the shared resource manager must push an event on the ResourceManagement event channel to notify others of this condition.

5.4.1.1.51 TravelRouteConsumer (Class)

This interface allows other CHART objects to register as a direct consumer of travel route statistical data. It provides operations for the travel route to call when the travel time or toll rate for the route is updated. A DMS registers as a TravelRouteConsumer when a TravelerInfoMsg is enabled.

5.4.1.1.52 TravInfoMsgSchedWatcherTask (Class)

5.4.1.1.53 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.4.1.2 DMSControlClassDiagram-ExternalDMS (Class Diagram)

This class diagram shows the relationship between classes that implement the ExternalDMS functionality.

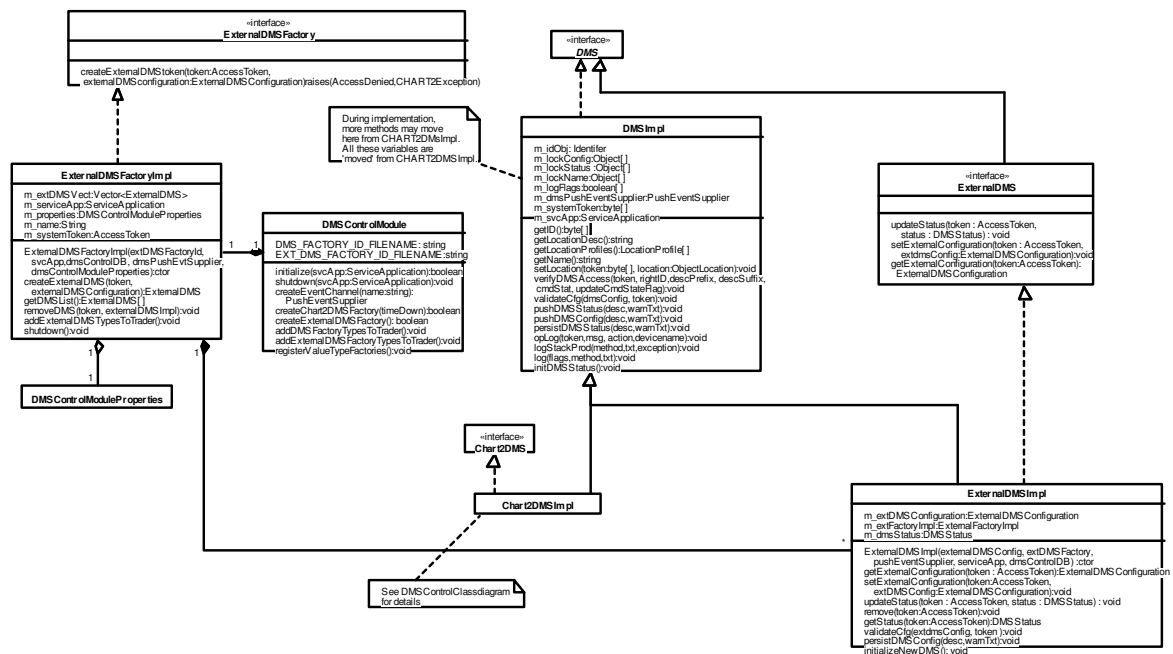


Figure 5-97 DMSControlClassDiagram-ExternalDMS (Class Diagram)

5.4.1.2.1 Chart2DMS (Class)

The Chart2DMS class extends the DMS interface and defines a more detailed interface to be used in manipulating the CHART-specific DMS objects within CHART. It provides an interface for traffic events to provide input as to what each traffic event desires to be on the sign via the ArbitrationQueue interface. Through the HARMMessageNotifier interface, a

HAR can use a DMS to notify travelers to tune in to a radio station to hear a traffic message. CHART business rules include concepts such as shared resources, arbitration queues, and linking device usage to traffic events. These concepts go beyond industry-standard DMS control. This includes an ability to enable and disable CHART traveler information messages, which were added in R3B3.

5.4.1.2.2 Chart2DMSImpl (Class)

The Chart2DMSImpl class provides an implementation of the Chart2DMS interface, and by extension the DMS, SharedResource, HARMMessageNotifier, CommEnabled, GeoLocatable, ArbitrationQueue and UniquelyIdentifiable interfaces, as specified by the IDL.

This class contains a CommandQueue object that is used to sequentially execute long running operations (field communications to the device) in a thread separate from the CORBA request threads, thus allowing quick initial responses. The Chart2DMSImpl also contains a MessageQueue, which is used by the ArbitrationQueue interface methods to handle requests from TrafficEvents to display or remove messages from the signs in online mode. When the Chart2DMSImpl evaluates its messages in the MessageQueue, it combines the highest priority messages into a single message which is placed into an appropriate QueueableCommand object (subclass of QueueableCommand) and added to the CommandQueue.

Also contained in this class are Chart2DMSConfiguration and Chart2DMSStatus objects (used to store the configuration and status of the sign), and a Chart2DMSData object (used to store internal status information which is persisted but not pushed out to clients), a list of ArbQueueEntry objects from the MessageQueue that are currently active on the sign, and a copy of the last QueueableCommand to put a message on the sign.

The Chart2DMSImpl contains *Impl methods that map to methods specified in the IDL, including requests to put a message on the sign or remove a message (in maintenance mode only), put the sign online, put the sign offline, put the sign in maintenance mode, or to change (set) the configuration of the sign. All of these requests require (or potentially require) field communications to the device, so each request is stored in a specific subclass of QueueableCommand and added to the CommandQueue. The queueable command objects simply call the appropriate Chart2DMSImpl method as the command is executed by the CommandQueue in its thread of execution.

The Chart2DMSImpl also contains methods called by the Chart2DMSFactory to support the timer tasks of the DMS Service: to poll the DMS devices, to look for DMS devices with communications timeout exceeded, to look for maintenance mode DMS devices with no one logged in at the controlling operations center, and to initiate recovery processing if needed.

5.4.1.2.3 DMS (Class)

The DMS class defines an interface to be used in manipulating Dynamic Message Sign (DMS) objects within Chart II. It specifies methods for setting messages and clearing

messages from a sign (in maintenance mode), polling a sign, changing the configuration of a sign, and resetting a sign. (Setting messages on a sign in online mode are not accomplished by manipulating a DMS directly; that is accomplished by manipulating traffic events, which use an ArbitrationQueue interface or by manipulating HARs, which use a HARMessageNotifier interface. This activity involves the DMS extension, Chart2DMS, which defines interactions with signs under Chart II business rules.)

5.4.1.2.4 DMSControlModule (Class)

The DMSControlModule class is the service module for the DMS devices and a DMS factory. It implements the ServiceApplicationModule interface. It creates and serves a single DMSFactoryImpl object, which in turn serves zero or more Chart2DMSImpl objects. It also creates DMSControlDB, DictionaryWrapper, DMSControlModuleProperties, and PushEventSupplier and NotificationChannel objects.

5.4.1.2.5 DMSControlModuleProperties (Class)

The DMSControlModuleProperties class is used to provide access to properties used by the DMS Control Module. This class wraps properties that are passed to it upon construction. It adds its own defaults and provides methods to extract properties specific to the DMS Control Module.

5.4.1.2.6 DMSImpl (Class)

This class implements the DMS interface. It provides methods to configure, control DMS objects in CHART

5.4.1.2.7 ExternalDMS (Class)

The ExternalDMS class extends the DMS interface and defines a more detailed interface to be used in manipulating the External DMS objects within CHART.

5.4.1.2.8 ExternalDMSFactory (Class)

The ExternalDMSFactory interface extends the DMSFactory interface.. This factory creates ExternalDMS objects (extensions of DMS objects).

5.4.1.2.9 ExternalDMSFactoryImpl (Class)

This class implements the ExternalDMSFactory interface. It provides the interface to create, remove and list ExternalDMS objects in CHART mirroring the data from external agencies.

5.4.1.2.10 ExternalDMSImpl (Class)

This class implements the ExternalDMS interface. It provides the interface to ExternalDMS objects in CHART

5.4.1.3 QueueableCommandClassDiagram (Class Diagram)

This class diagram shows the classes derived from QueueableCommand necessary for DMS Control. A class exists for each type of command that can be executed asynchronously on a DMS object.

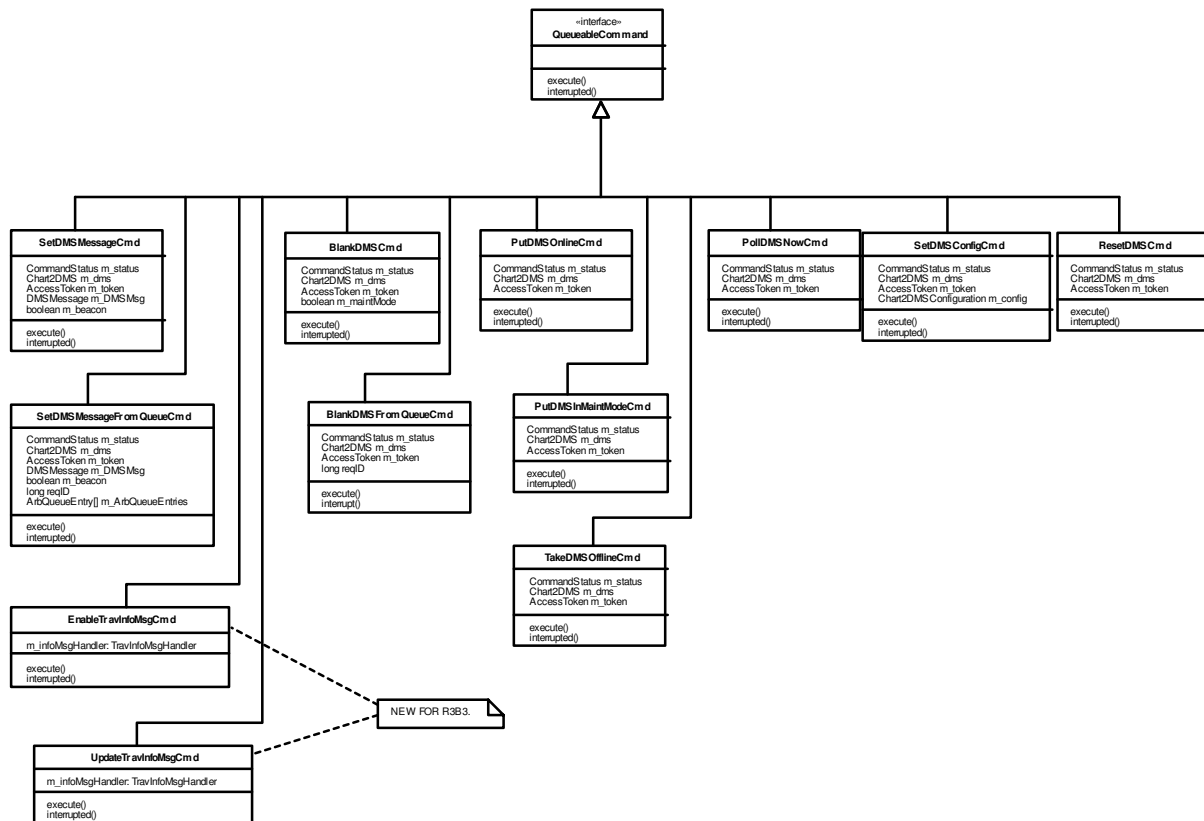


Figure 5-98 QueueableCommandClassDiagram (Class Diagram)

5.4.1.3.1 BlankDMSCmd (Class)

The BlankDMSCmd class is a QueueableCommand subclass which contains data necessary to send a request to a Chart2DMSImpl to blank the sign in maintenance mode. It is created by the Chart2DMSImpl during successful processing of its blankSign and deactivateHARNotice methods. When the CommandQueue invokes the execute method of this class, it merely calls the blankSignImpl method of the appropriate Chart2DMSImpl object with the data stored within this class.

5.4.1.3.2 BlankDMSFromQueueCmd (Class)

The BlankDMSFromQueueCmd class is a QueueableCommand subclass which contains data necessary to send a request to a Chart2DMSImpl to blank the sign during normal operations (online mode). It is created by the Chart2DMSImpl during successful

processing of its evaluateQueue method. When the CommandQueue invokes the execute method of this class, it merely calls the blankSignFromQueueImpl method of the appropriate Chart2DMSImpl object with the data stored within this class.

5.4.1.3.3 EnableTravInfoMsgCmd (Class)

The EnableTravInfoMsgCmd class is a QueueableCommand subclass which contains data necessary to send a request to a Chart2DMSImpl to activate Travel Info Message a message on the sign. It is created by the DMSTravInfoHandler during setTravInfoMsgEnabledFlag call.

5.4.1.3.4 PollDMSNowCmd (Class)

The PollDMSNowCmd class is a QueueableCommand subclass which contains data necessary to send a request to a Chart2DMSImpl to poll its device. It is created by the Chart2DMSImpl during successful processing of its pollNow method in maintenance mode (triggered by a user request) or during processing of the pollIfNecessary method (triggered by the automatic polling of the PollDMSTask object). When the CommandQueue invokes the execute method of this class, it merely calls the pollNowImpl method of the appropriate Chart2DMSImpl object with the data stored within this class.

5.4.1.3.5 PutDMSInMaintModeCmd (Class)

The PutDMSInMaintModeCmd class is a QueueableCommand subclass which contains data necessary to send a request to a Chart2DMSImpl to put the sign in maintenance mode (from either offline or online mode). It is created by the Chart2DMSImpl during successful processing of its putInMaintMode method. When the CommandQueue invokes the execute method of this class, it merely calls the putInMaintModeImpl method of the appropriate Chart2DMSImpl object with the data stored within this class.

5.4.1.3.6 PutDMSOnlineCmd (Class)

The PutDMSOnlineCmd class is a QueueableCommand subclass which contains data necessary to send a request to a Chart2DMSImpl to put the sign online (from either offline or maintenance mode). It is created by the Chart2DMSImpl during successful processing of its putDMSOnline method. When the CommandQueue invokes the execute method of this class, it merely calls the putDMSOnlineImpl method of the appropriate Chart2DMSImpl object with the data stored within this class.

5.4.1.3.7 QueueableCommand (Class)

A QueueableCommand is an interface used to represent a command that can be placed on a CommandQueue for asynchronous execution. Derived classes implement the execute method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a CommandQueue. The CommandQueue driver calls the execute method to execute a command in the queue and a call to the interrupted method is made when a CommandQueue is shut down.

5.4.1.3.8 ResetDMSCmd (Class)

The ResetDMSCmd class is a QueueableCommand subclass which contains data necessary to send a request to a Chart2DMSImpl to put reset the sign (in maintenance mode only). It is created by the Chart2DMSImpl during successful processing of its resetController method. When the CommandQueue invokes the execute method of this class, it merely calls the resetControllerImpl method of the appropriate Chart2DMSImpl object with the data stored within this class.

5.4.1.3.9 SetDMSConfigCmd (Class)

The SetDMSConfigCmd class is a QueueableCommand subclass which contains data necessary to send a request to a Chart2DMSImpl to update its configuration (in maintenance mode only). It is created by the Chart2DMSImpl during successful processing of its setConfiguration method. When the CommandQueue invokes the execute method of this class, it merely calls the setConfigurationImpl method of the appropriate Chart2DMSImpl object with the data stored within this class.

5.4.1.3.10 SetDMSMessageCmd (Class)

The SetDMSMessageCmd class is a QueueableCommand subclass which contains data necessary to send a request to a Chart2DMSImpl to put a message on the sign in maintenance mode. It is created by the Chart2DMSImpl during successful processing of its setMessage and activateHARNotice methods. When the CommandQueue invokes the execute method of this class, it merely calls the setMessageImpl method of the appropriate Chart2DMSImpl object with the data stored within this class.

5.4.1.3.11 SetDMSMessageFromQueueCmd (Class)

The SetDMSMessageFromQueueCmd class is a QueueableCommand subclass which contains data necessary to send a request to a Chart2DMSImpl to put a message on the sign during normal operations (online mode). It is created by the Chart2DMSImpl during successful processing of its setMessageFromQueue and evaluateQueue methods. When the CommandQueue invokes the execute method of this class, it merely calls the setDMSMessageFromQueueImpl method of the appropriate Chart2DMSImpl object with the data stored within this class.

5.4.1.3.12 TakeDMSOfflineCmd (Class)

The TakeDMSOfflineCmd class is a QueueableCommand subclass which contains data necessary to send a request to a Chart2DMSImpl to put the sign offline (from either online or maintenance mode). It is created by the Chart2DMSImpl during successful processing of its takeOffline method. When the CommandQueue invokes the execute method of this class, it merely calls the takeOfflineImpl method of the appropriate Chart2DMSImpl object with the data stored within this class.

5.4.1.3.13 UpdateTravInfoMsgCmd (Class)

TheUpdateTravInfoMsgCmd class is a QueueableCommand subclass which contains data necessary to send a request to a Chart2DMSImpl to activate Travel Info Message a message on the sign. It is created by the DMSTravInfoHandler during checkMessage call.

5.4.2 Sequence diagrams

5.4.2.1 CHART2DMSImpl:validate (Sequence Diagram)

This Sequence Diagram shows how a Chart2DMSImpl object responds to request validate DMSTravInfoMsgHandler. This method return true if ArbQueueEntryKey passed in exist in one of the DMSTravInfoMsgHandlerobjects owned by this Chart2DMSImpl.

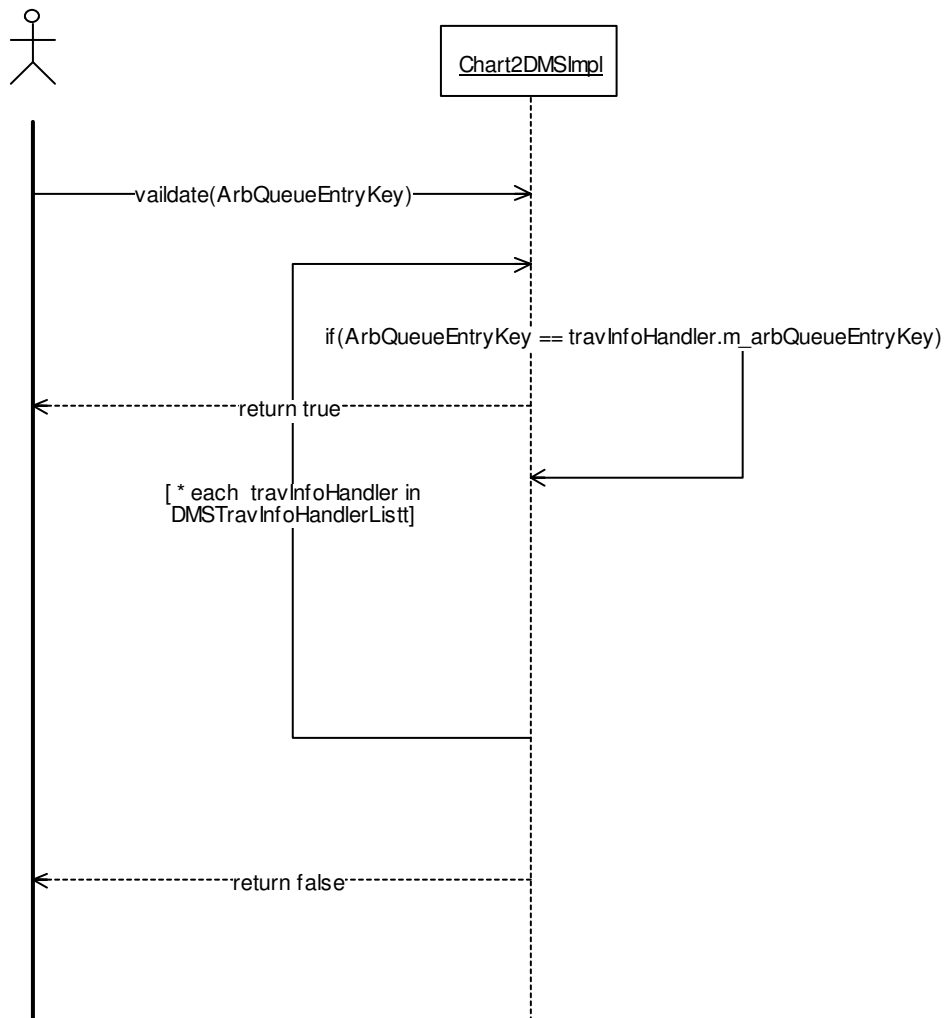


Figure 5-99 CHART2DMSImpl:validate (Sequence Diagram)

5.4.2.2 Chart2DMSFactoryImpl:checkTravInfoMsgSchedule (Sequence Diagram)

This Sequence Diagram shows how a Chart2DMSImpl checks TravInfoMsgSchedule.Timer calls run() on TravInfoMsgSchedWatcherTask with call checkTracInfoMsgSchedEnabled on each DMS.

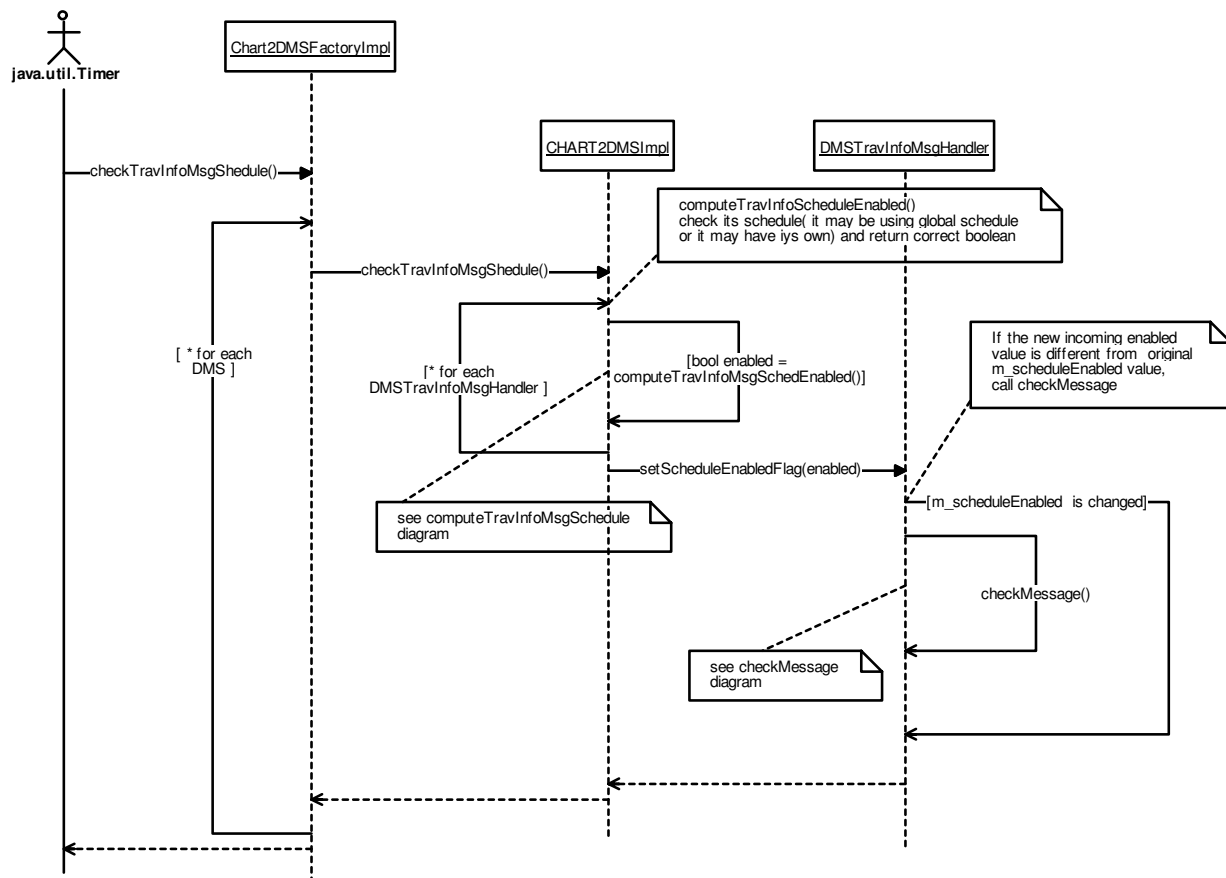


Figure 5-100 Chart2DMSFactoryImpl:checkTravInfoMsgSchedule (Sequence Diagram)

5.4.2.3 Chart2DMSImpl:RouteUpdate (Sequence Diagram)

This Sequence Diagram shows how a Chart2DMSImpl object responds to request update routes status data. The requesting operator must have Chart2System right. This diagram include calls to RouteTravelTimeStatsUpdated, RouteTollRateStatsUpddated, RoteUpdateCompleted amd RouteConfigUpdated methods on Chart2DMSImpl.

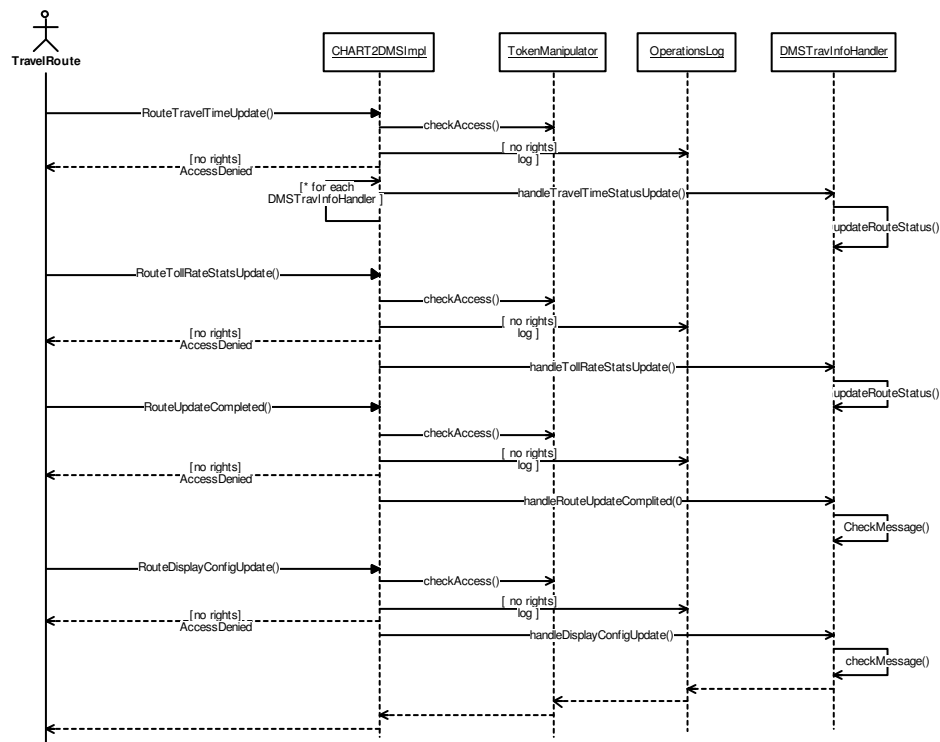


Figure 5-101 Chart2DMSImpl:RouteUpdate (Sequence Diagram)

5.4.2.4 Chart2DMSImpl:addDMSTravInfoMsg (Sequence Diagram)

This Sequence Diagram shows how a Chart2DMSImpl responds to a request to add TravInfoMsg to DMS TravInfoMsg config. Requesting operator must have proper functional rights.

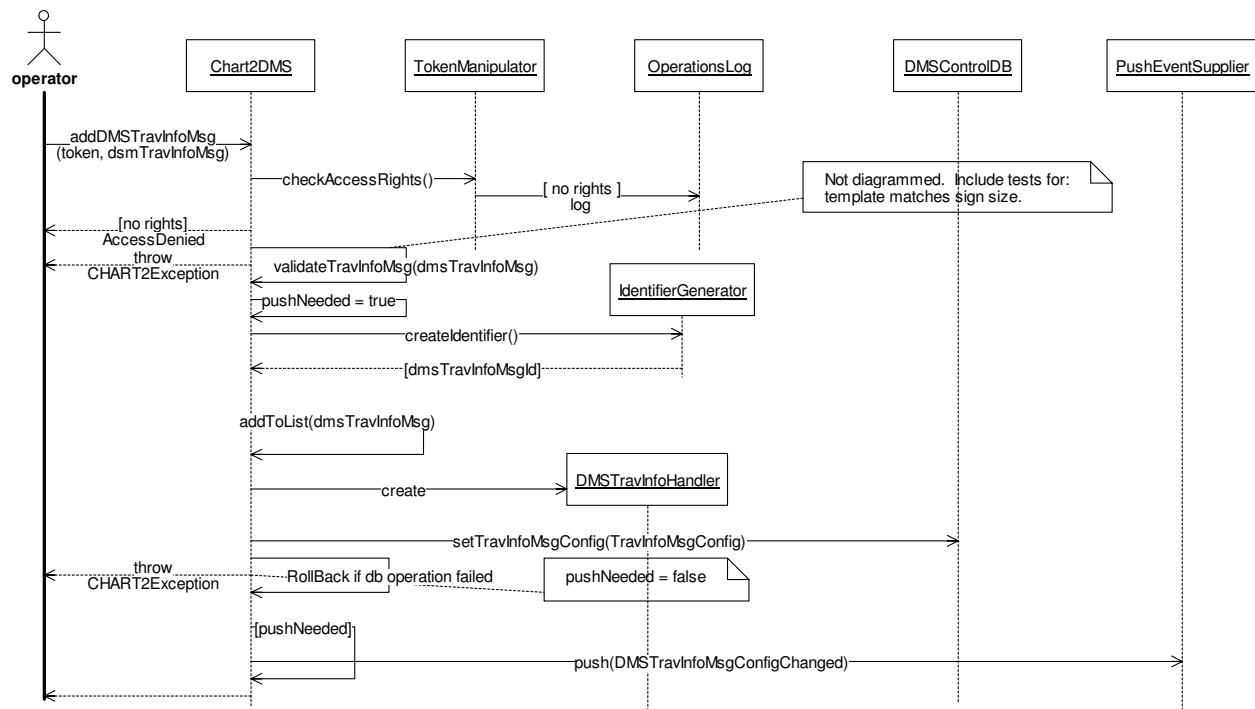


Figure 5-102 Chart2DMSImpl:addDMSTravInfoMsg (Sequence Diagram)

5.4.2.5 Chart2DMSImpl:computeTravInfoMsgSchedEnabled (Sequence Diagram)

This Sequence Diagram shows how a Chart2DMSImpl checks the DMS's schedule(it may be using global schedule or it have its own), compare to current time of the day, and return the correct boolean.

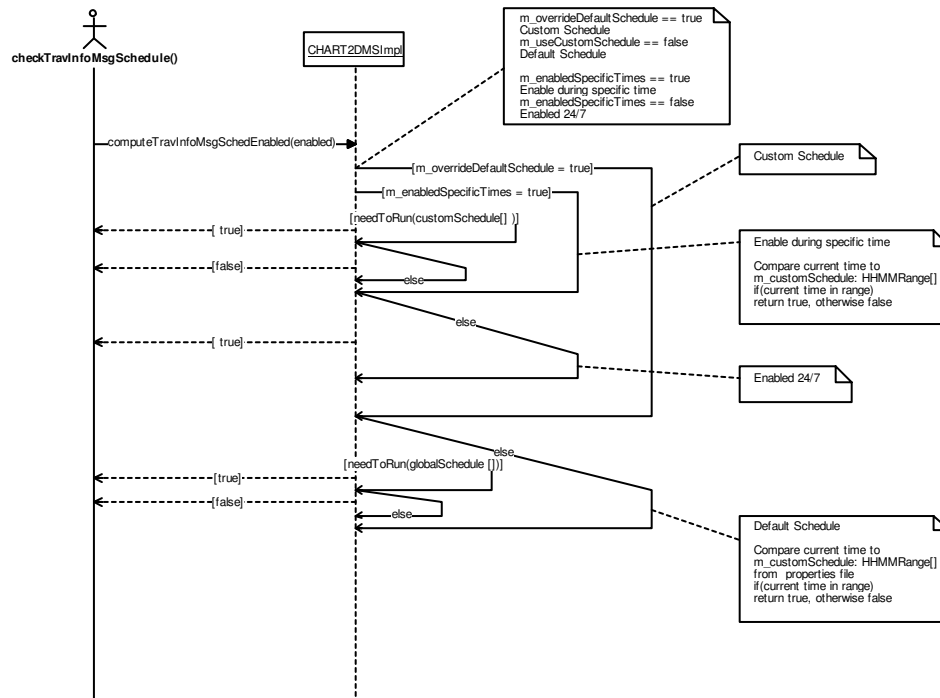


Figure 5-103 Chart2DMSImpl:computeTravInfoMsgSchedEnabled (Sequence Diagram)

5.4.2.6 Chart2DMSImpl:modifyDMSTravInfoMsg (Sequence Diagram)

This Sequence Diagram shows how a Chart2DMSImpl responds to a request to modify TravInfoMsg in DMS TravInfoMsg config. Requesting operator must have proper functional rights.

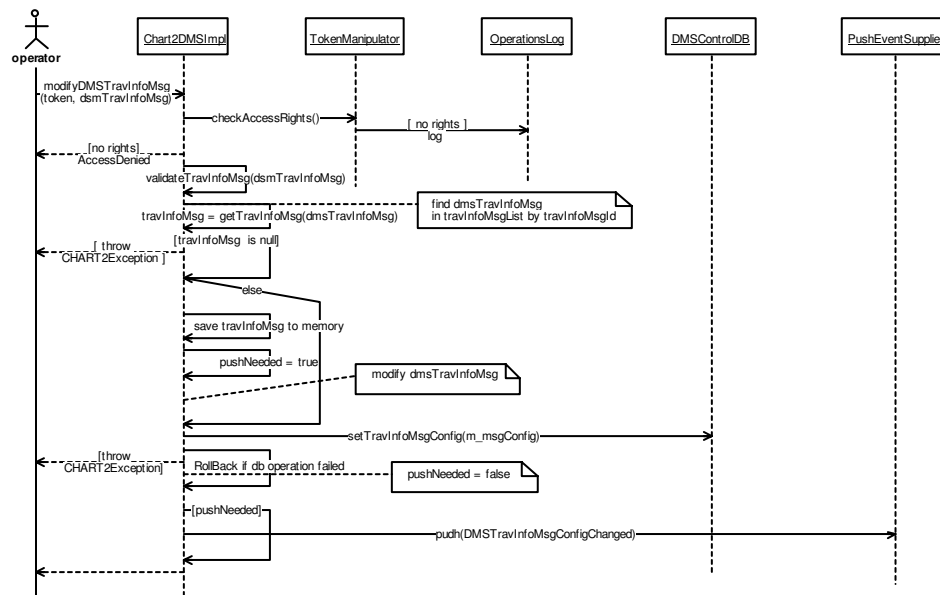


Figure 5-104 Chart2DMSImpl:modifyDMSTravInfoMsg (Sequence Diagram)

5.4.2.7 Chart2DMSImpl:removeDMSTravInfoMsg (Sequence Diagram)

This Sequence Diagram shows how a Chart2DMSImpl responds to a request to remove TravInfoMsg from DMS TravInfoMsg config. Requesting operator must have proper functional rights.

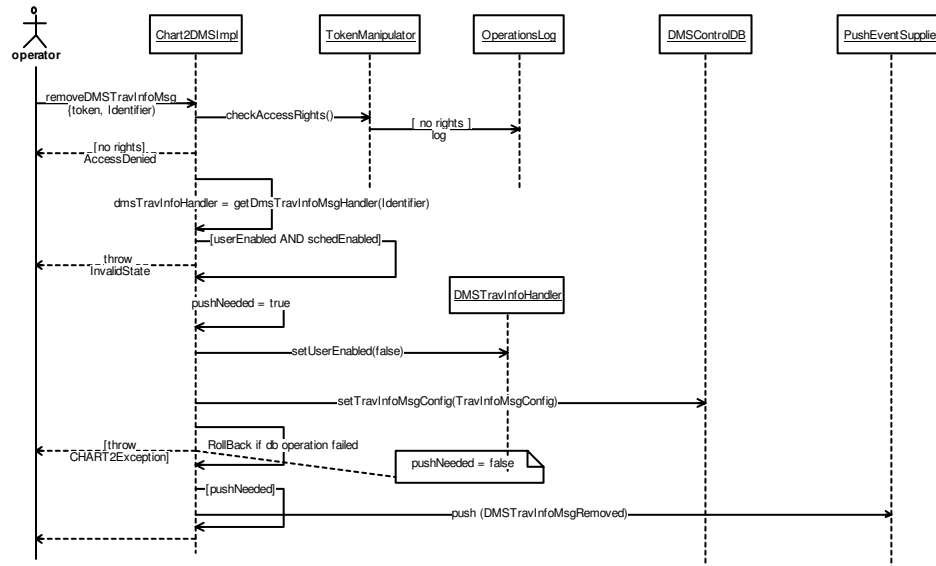


Figure 5-105 Chart2DMSImpl:removeDMSTravInfoMsg (Sequence Diagram)

5.4.2.8 Chart2DMSImpl:routeDeleted (Sequence Diagram)

This Sequence Diagram shows how a Chart2DMSImpl object responds to request route deleted. This method will set the routeId in any TravelerInfoMsg to NULL identifier and also in any relatedRoutes in TravelerInfoMsgConfig.

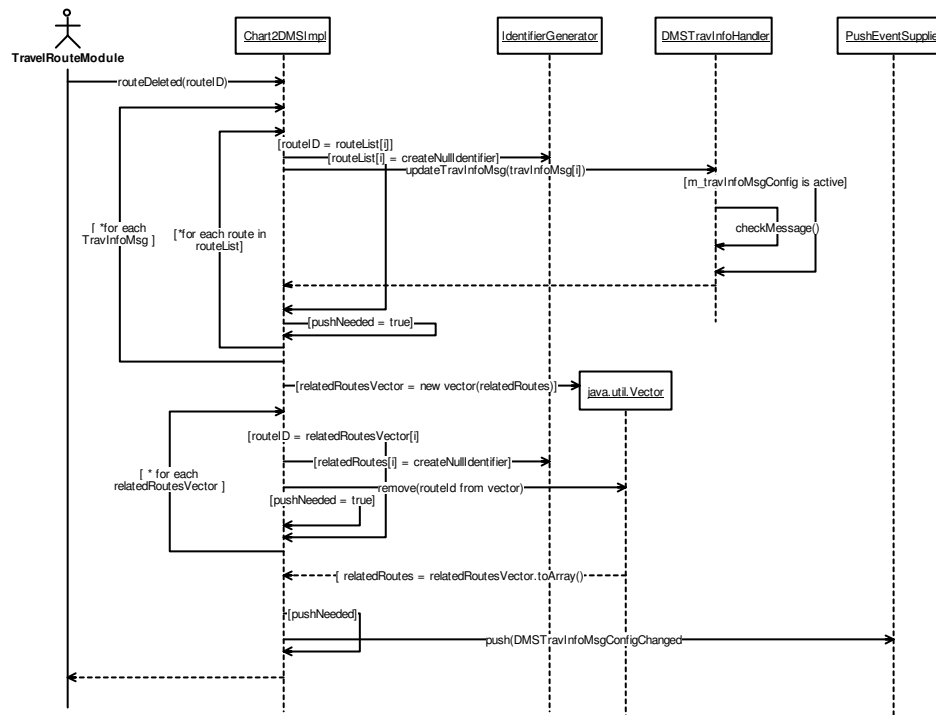


Figure 5-106 Chart2DMSImpl:routeDeleted (Sequence Diagram)

5.4.2.9 Chart2DMSImpl:setQueueLevels (Sequence Diagram)

This Sequence Diagram shows how a Chart2DMSImpl responds to a request to change the set QueueLevels on configuration of a DMS.

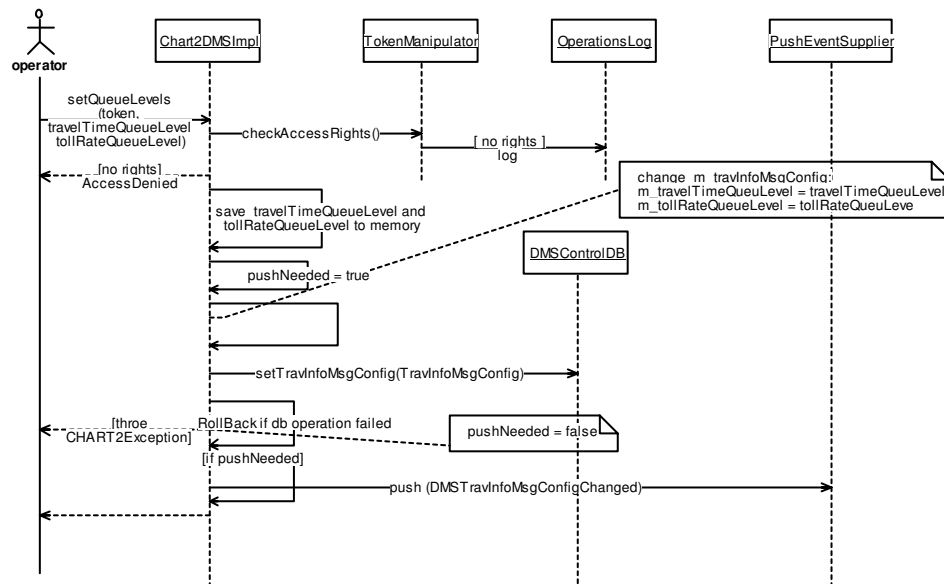


Figure 5-107 Chart2DMSImpl:setQueueLevels (Sequence Diagram)

5.4.2.10 Chart2DMSImpl:setRelatedRoutes (Sequence Diagram)

This Sequence Diagram shows how a Chart2DMSImpl responds to a request to set RelatedRoutes in DMS TravInfoMsg config. Requesting operator must have proper functional rights.

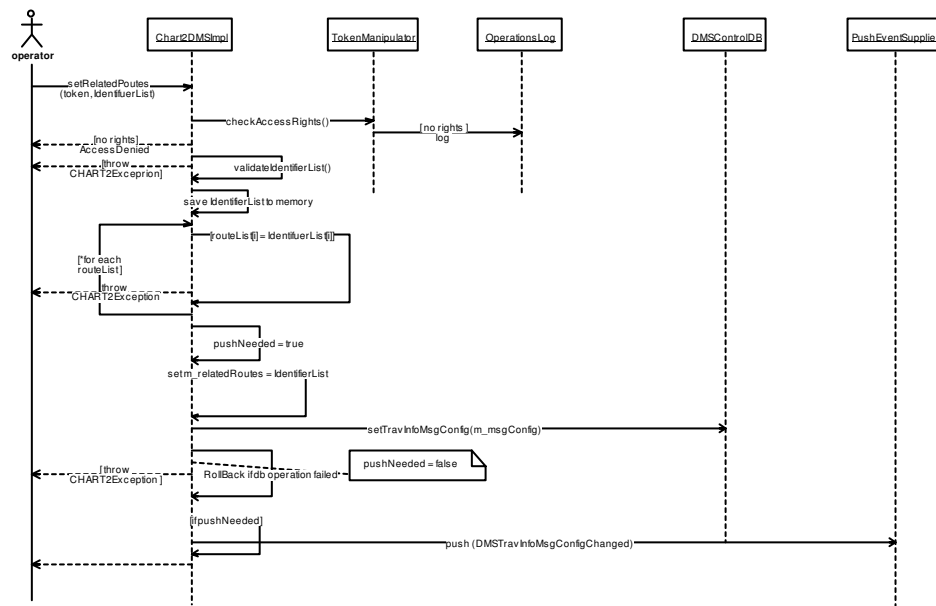


Figure 5-108 Chart2DMSImpl:setRelatedRoutes (Sequence Diagram)

5.4.2.11 Chart2DMSImpl:setTravInfoMsgEnabledFlag (Sequence Diagram)

This Sequence Diagram shows how a DMSTravInfoMsgHandler responds to request set TravInfoMsg enabled flag. The requesting operator must have proper functional rights. This method calls setUserEnabled on all DMSTravInfoMsgHandler objects, if DMSTravInfoMsgHandler was enabled DMSTravInfoMsgHandler call each its routes removeConsumer method. Then its construct an EnableTravInfoMsgCmd and queue on DMSImpl own CommandQueue

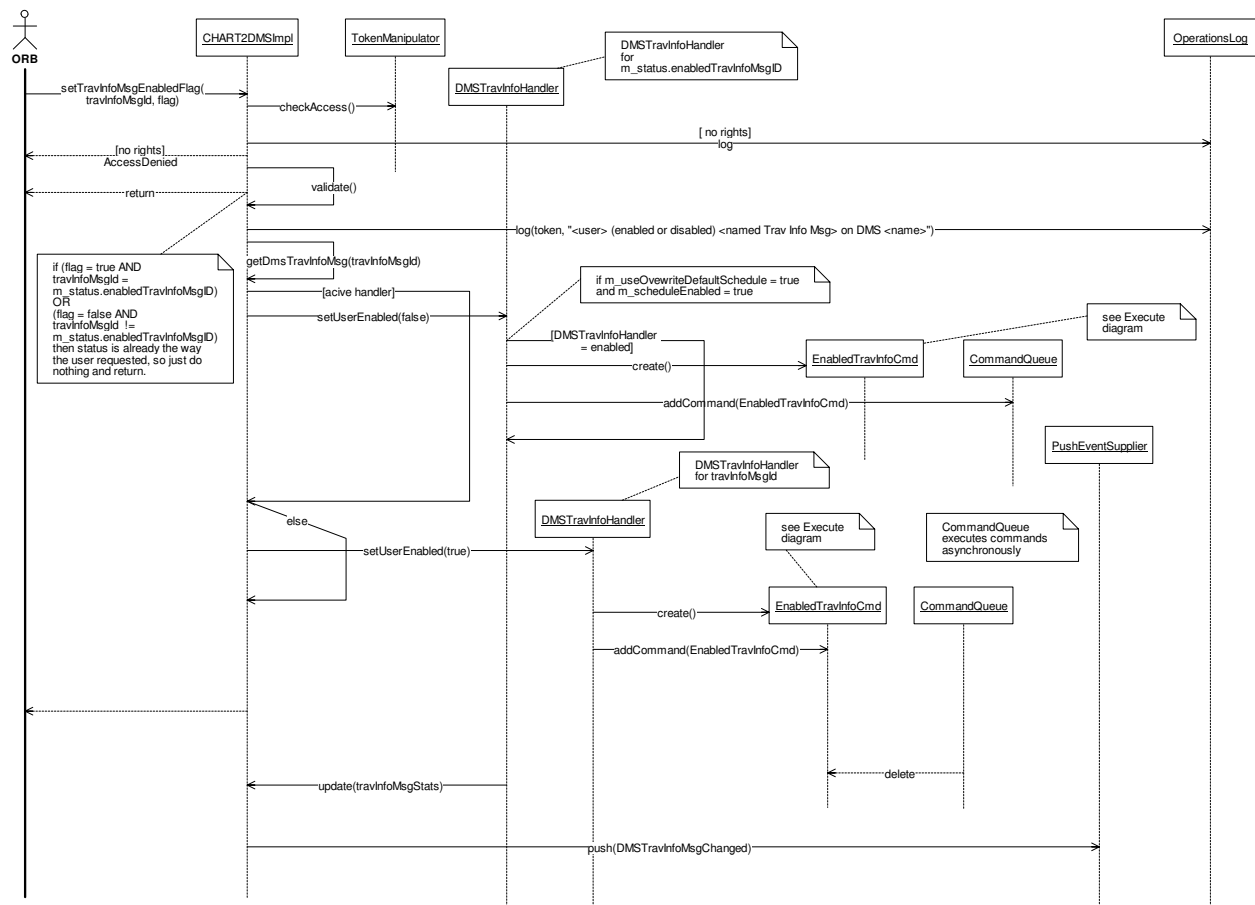


Figure 5-109 Chart2DMSImpl:setTravInfoMsgEnabledFlag (Sequence Diagram)

5.4.2.12 Chart2DMSImpl:setTravelTimeSchedule (Sequence Diagram)

This Sequence Diagram shows how a Chart2DMSImpl responds to a request to change the set TrvelTimeSchedule on configuration of a DMS.

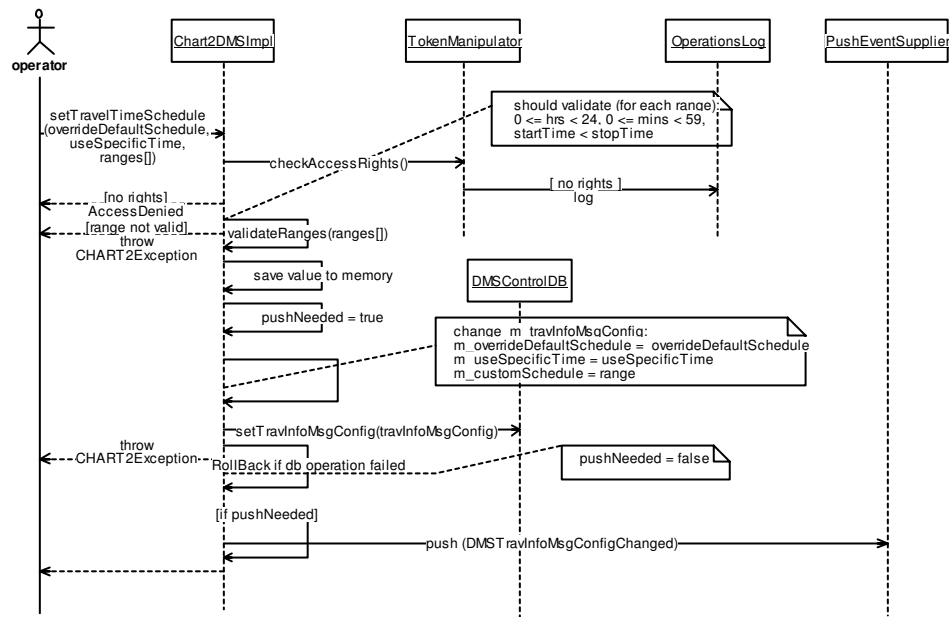


Figure 5-110 Chart2DMSImpl:setTravelTimeSchedule (Sequence Diagram)

5.4.2.13 DMSControlModule:FmsGetConnectedPort (Sequence Diagram)

This sequence diagram shows how a DMSImpl object gets a connected port. This method is called from several other methods in the DMS service. A modem port is obtained from the ModemPortLocator object. On failure, a call is made to the helper method handleOpStatus to deal with the case where the operational status has changed. The command status is either updated or completed during the call to the ModemPortLocator object based on a flag passed into this method.

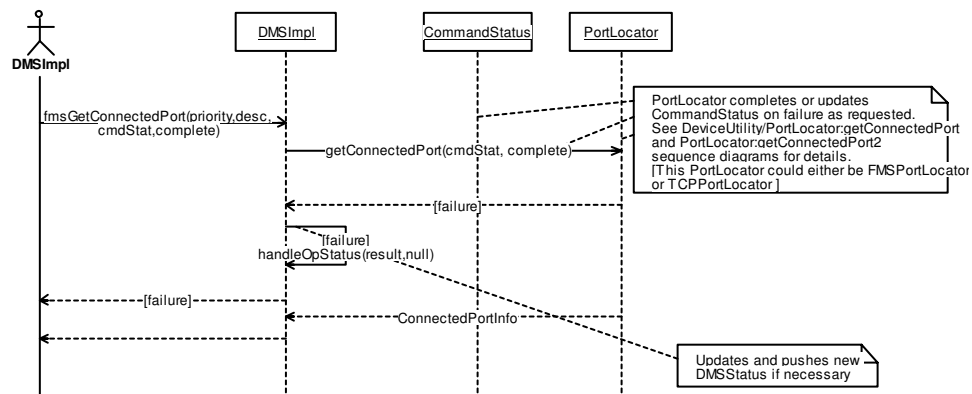


Figure 5-111 DMSControlModule:FmsGetConnectedPort (Sequence Diagram)

5.4.2.14 DMSControlModule:FmsReleasePort (Sequence Diagram)

This helper method releases an FMS port which is no longer needed. It disconnects the port first, then calls the PortLocator to release it. Errors are logged, but not reported, as the port will be released or reclaimed in any case, and errors relating to releasing a port would mask an otherwise successful status or more a useful error status.

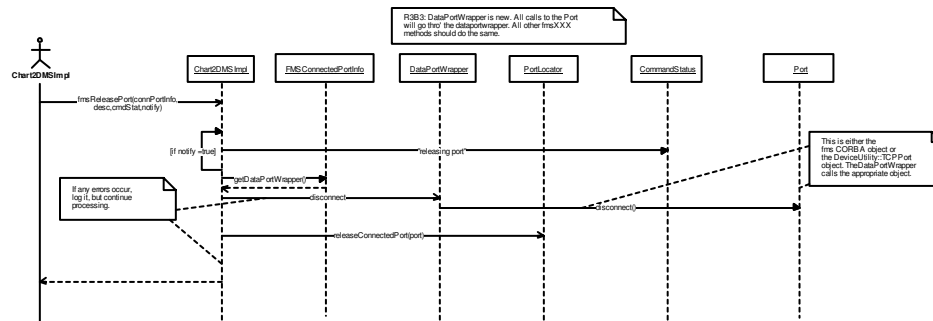


Figure 5-112 DMSControlModule:FmsReleasePort (Sequence Diagram)

5.4.2.15 DMSControlModule:HandleOpStatus (Sequence Diagram)

This Sequence Diagram shows how a Chart2DMSImpl handles the task of detecting and responding to changes in its operational status (whether it is in "OK", "COMM_FAILURE" or "HARDWARE_FAILURE" status). A DMS is normally "OK", but falls into "COMM_FAILURE" when FMS reports that it cannot communicate with the device, and into "HARDWARE_FAILURE" when the FMS can communicate with the device but the device or FMS is detecting some sort of hardware problem with the device itself. At this point, HARDWARE_FAILURE and COMM_FAILURE are treated virtually identically. This method is called, with the status reported back from FMS, after every attempt to communicate with the device, and processing falls into one of three cases, depending on the status reported (although the two failure cases are nearly identical).

If the device now being reported OK and it was already OK, there is no change in status, and all that is necessary is to update the m_lastContactTime of the device. (This variable is used to determine when to poll [see runPollDMSTask] and when to declare that a "Communications Timeout" has occurred [see runCheckCommLossTask].) If the status has just become OK, this fact is logged, and the new DMSStatus is persisted and pushed out into the event channel. The DMS is polled to determine its current status. If the device is online, and m_needsReevaluation is true, this means an earlier attempt to set the device to the correct condition (new message, default message) has failed since the device went COMM_FAILED, so evaluateQueue is called to ensure that the correct message is put on the DMS.

If the device is now being reported with a failure and the device was already in that failure condition, there is no change in status, and nothing is done. If the status is just now changing, this is logged, the DMSStatus is persisted and pushed out into the event channel, and a device failure alert is created. Note that if the device has gone into COMM_FAILURE, and it remains in this condition for the timeout period, the CheckCommLossTask's run method will detect and handle it (see runCheckCommLossTask). Until the timeout period expires, it is assumed that the message is still on the sign, so no further action is taken now. If the device has gone into HARDWARE_FAILURE, FMS is still in contact with it, and changes in status (e.g., loss of a message) can be detected by other means, for instance, by polling (see runPollDMSTask).

A DeviceFailure Alert is only created when the DMS transitions into HARDWARE_FAILURE. Any future transitions into another state have no effect on the alert.

When a comm failure is detected, a DeviceFailureAlert is created. Also Notifications are sent when HARDWARE_FAILURE or COMM_FAILURE happens.

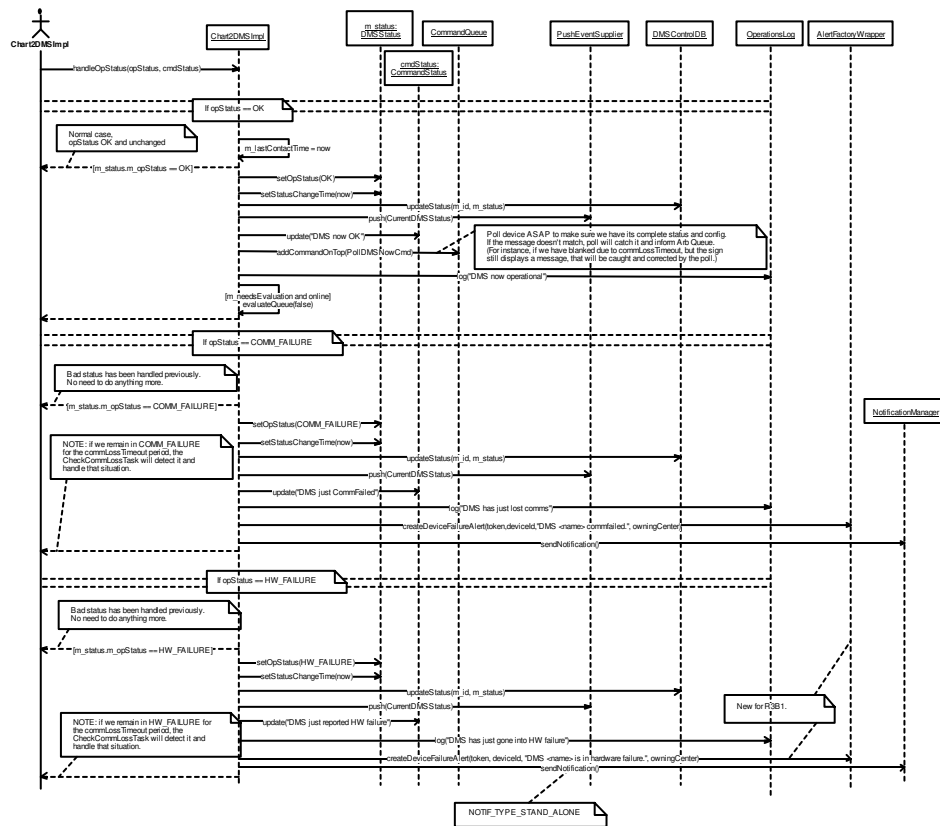


Figure 5-113 DMSControlModule:HandleOpStatus (Sequence Diagram)

5.4.2.16 DMSControlModule:Initialize (Sequence Diagram)

This Sequence Diagram shows how the DMSControlModule is started. This module is created by a service application that will host this module's objects. A ServiceApplication is passed to this module's initialize method and provides access to basic objects needed by this module. This module creates a DMSFactory, which creates the known DMS objects which have been persisted into the database. Two PushEventSupplier objects, one for status, configuration, and existence changes and one for abandoned DMSs (active DMSs with no one logged in at the controlling operations center), are created. In addition, NotificationChannel and DMSControlDB objects are created.

The DMSFactory and DMS objects are published via the CORBA Trading Service to make them available for general status updates and as candidates for control (given the proper access rights). In addition, this service also performs regularly recurring maintenance functions controlled by timer tasks started by this initialize method.

the DMSControlModule also creates the ExternalDMSFactory and ExternalDMS objects that import DMS data from external agencies.

5.4.2.17 DMSControlModule:RestoreDMS (Sequence Diagram)

This Sequence Diagram shows how a DMSImpl is initialized (whether being depersisted or created from scratch). DMSProtocolHdlr, ModemPortLocator, CommandQueue, and MessageQueue objects are created. If the DMS is being depersisted, after the MessageQueue is depersisted, the MessageQueue method validateEntries() is called to attempt to contact the TrafficEvent IDs on the list to validate their existence. If not in recovery mode, this is the only chance the TrafficEvents get. If still within the recovery mode, another attempt to contact the traffic events will be made when the recovery period is over. This diagram also shows a summary of what happens when an entry is added to or reprioritized in the message queue during recovery mode, and what happens when the recovery mode period expires.

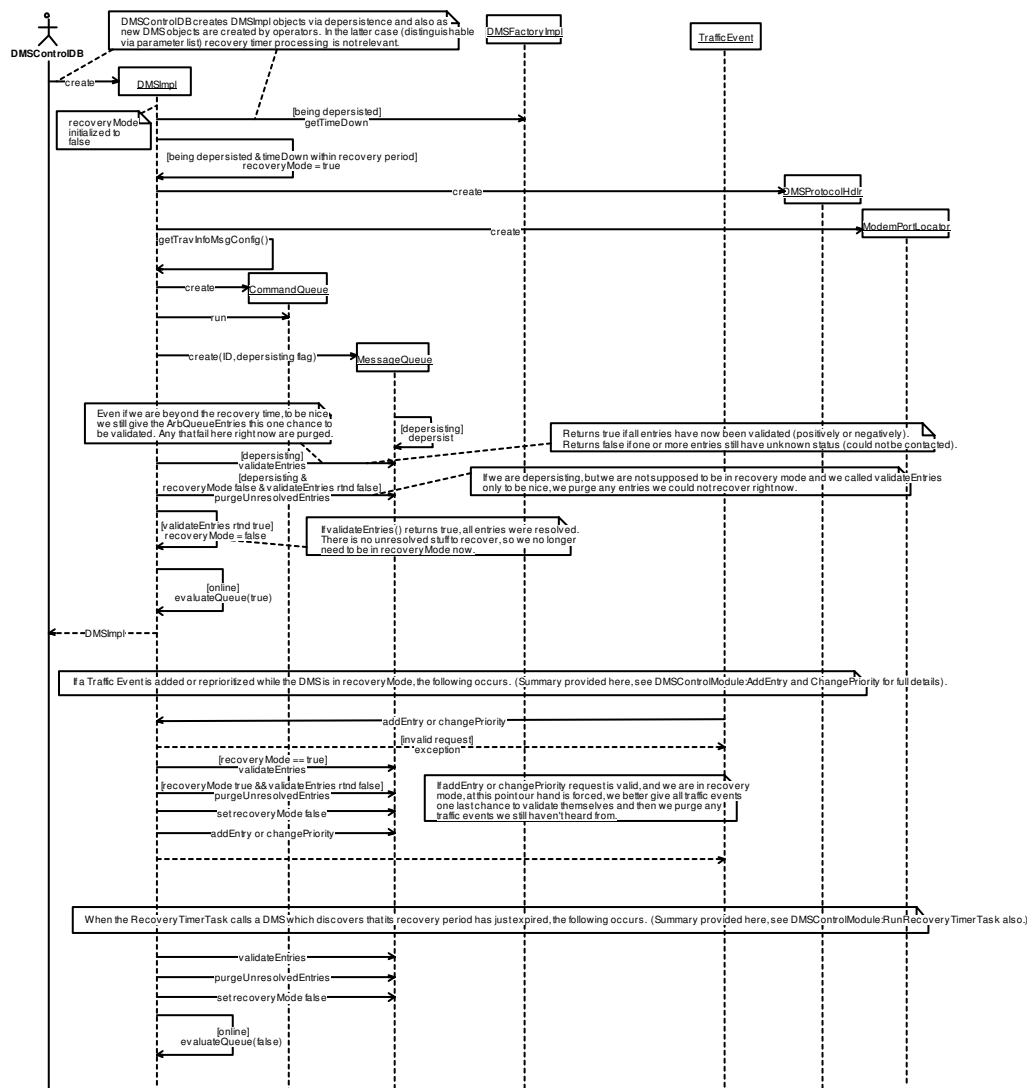


Figure 5-115 DMSControlModule:RestoreDMS (Sequence Diagram)

5.4.2.18 DMSControlModule:SetConfiguration (Sequence Diagram)

This Sequence Diagram shows how a Chart2DMSImpl responds to a request to change the configuration of a DMS. The DMS must be in maintenance mode, the requesting operator must have proper functional rights, and if there is a (maintenance mode) message on the sign from another operations center, the user must have override authority. This method creates a SetDMSConfigCmd (a QueueableCommand) and adds it to the DMS's CommandQueue. The CommandQueue is required since some configuration changes require field communications to the sign, and field communications are relatively slow and can queue up. Requests to communicate with the sign are processed on a first-come, first-served basis. When the CommandQueue is ready, it executes the SetDMSConfigCmd, which calls the setConfigurationImpl method, also shown on this diagram. When the setConfigurationImpl method runs, it checks that the DMS is still in maintenance mode (a previously queued command could have changed it), and that there is no resource conflict (a previously queued command could have written a message from an operator at another operations center). Assuming no problems, the Chart2DMSConfiguration is locked down, and all parameters which need to change are changed. If any of these parameter changes require communications to the sign (e.g., setting the Comm Loss Timeout in an FP9500), a new PortLocator is created using the new parameters. Then, FMS is requested to make the specified change(s). The method handleOpStatus handles and responds to any changes to the operational status of the sign (OK, comms failure, or hardware failure) reported by FMS during this operation. The new configuration is persisted and pushed into the event channel. The requesting user is kept abreast of progress of the request all the while, via a CommandStatus object viewable by the user.

5.4.2.19 DMSControlModule:Shutdown (Sequence Diagram)

This Sequence Diagram shows how the DMSControlModule is terminated. The DMSControlModule is shut down by the ServiceApplication that started it. When told to shut down, the DMSControlModule disconnects the DMSFactory from the ORB, withdraws its offer from the trader, and shuts down the object. When the DMSFactory is shut down, it withdraws the offers of each DMS, disconnects each DMS from the ORB, and shuts down each Chart2DMSImpl. Chart2DMSImpl shutdown processing includes destroying the MessageQueue and shutting down the CommandQueue. No information needs to be persisted to the database during shutdown, as information is written to the database as it is updated. When the ExternalDMSFactory is shutdown, it deactivates and deletes the ExternalDMS objects from memory.

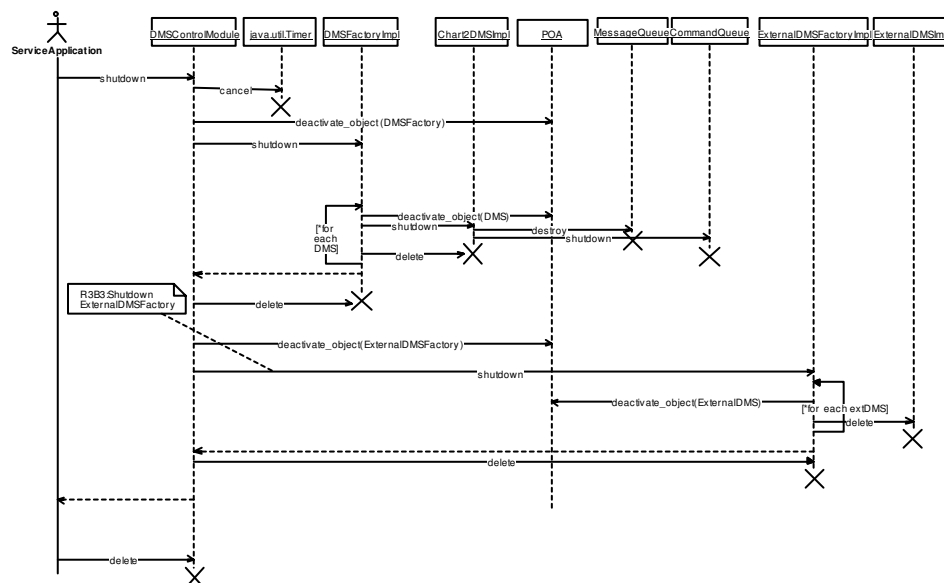


Figure 5-117 DMSControlModule:Shutdown (Sequence Diagram)

5.4.2.20 DMSImpl:setLocation (Sequence Diagram)

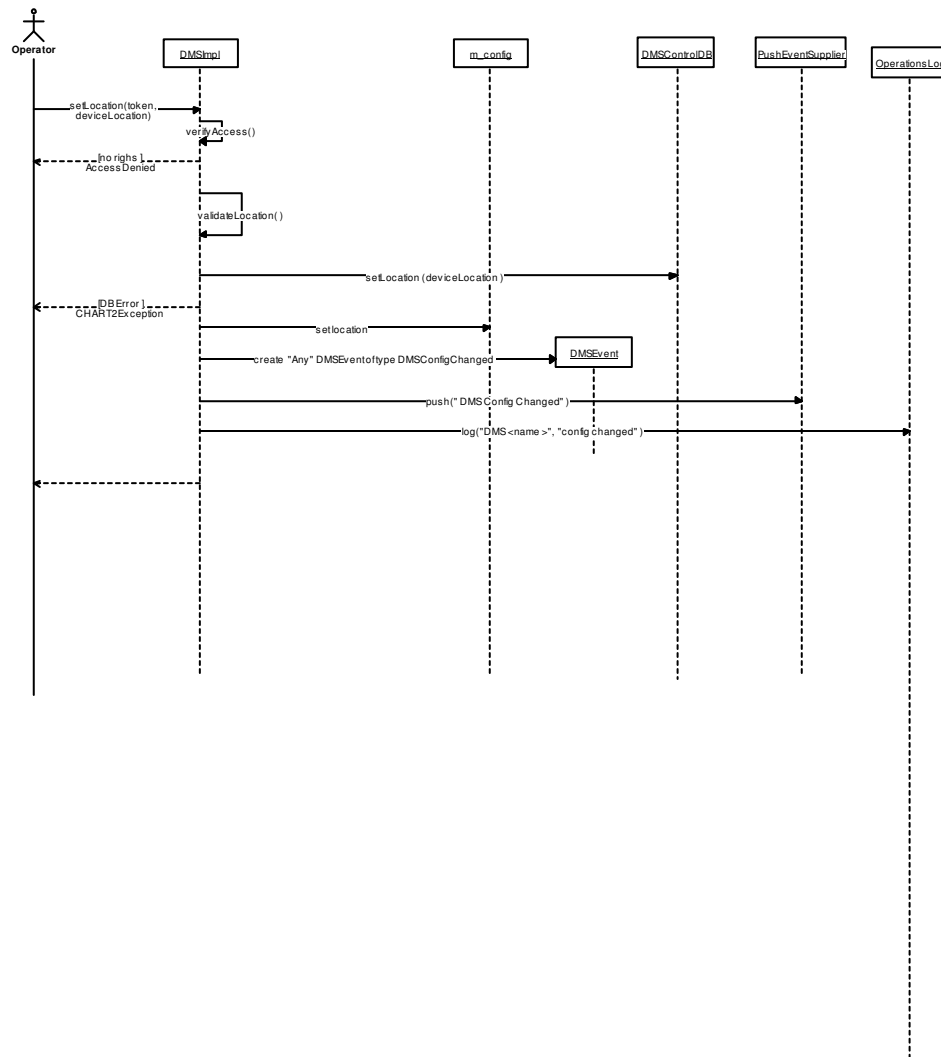


Figure 5-118 DMSImpl:setLocation (Sequence Diagram)

5.4.2.21 DMSTravInfoMsgHandler:checkMessage (Sequence Diagram)

This Sequence Diagram shows how a DMSTravInfoMsgHandler responds to request checkMessage. DMSTravInfoMsgHandler call checkMessage on itself. checkMessage queues a UpdateTravInfoMsgCmd on dms queue.

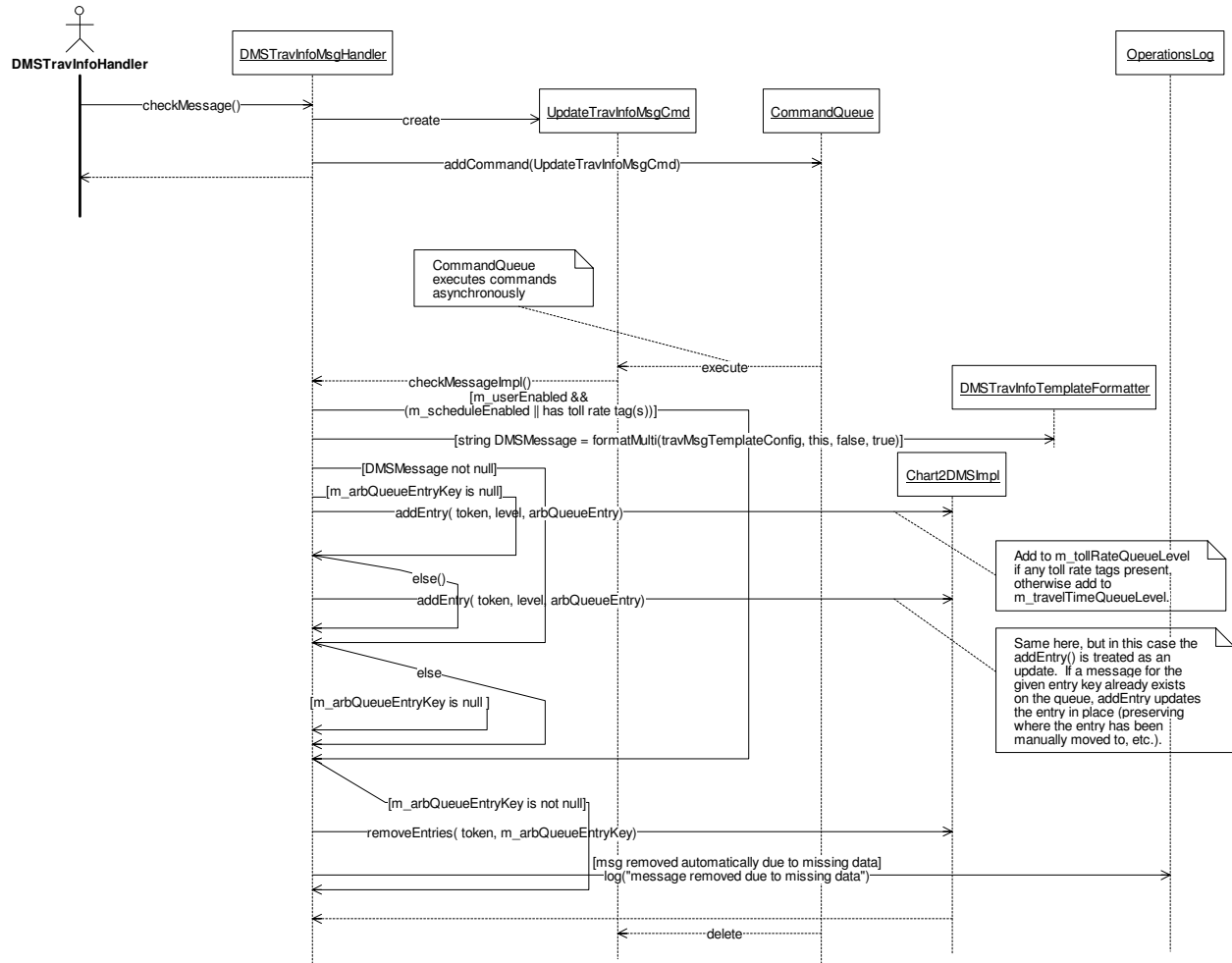


Figure 5-119 DMSTravInfoMsgHandler:checkMessage (Sequence Diagram)

5.4.2.2 DMSTravInfoMsgDataSupplier:getData (Sequence Diagram)

This Sequence Diagram shows how a DMSTravInfoMsgDataSupplier object responds to data requests for a DMS Traveler Information Message.

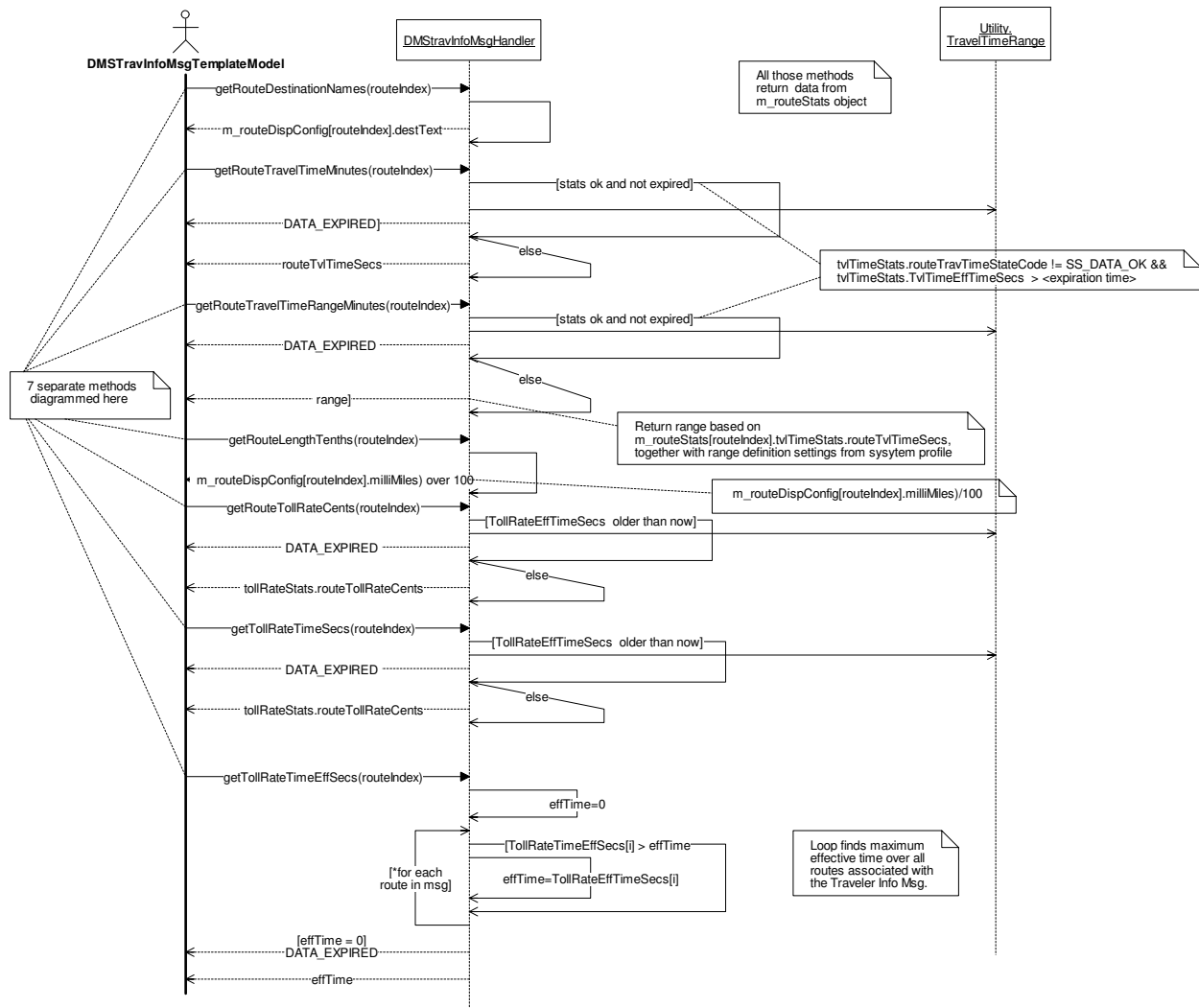


Figure 5-120 DMSTravInfoMsgDataSupplier:getData (Sequence Diagram)

5.4.2.23 EnabledTravInfoMsgCmd:execute (Sequence Diagram)

This Sequence Diagram shows how the EnabledTravInfoMsgCmd object executes its task when directed to run by the Java timer object. The run method of EnabledTravInfoMsgCmd calls setUserEnabled(true) on DMSTravInfoMsgHandler which calls addConsumer() on each of its routes.

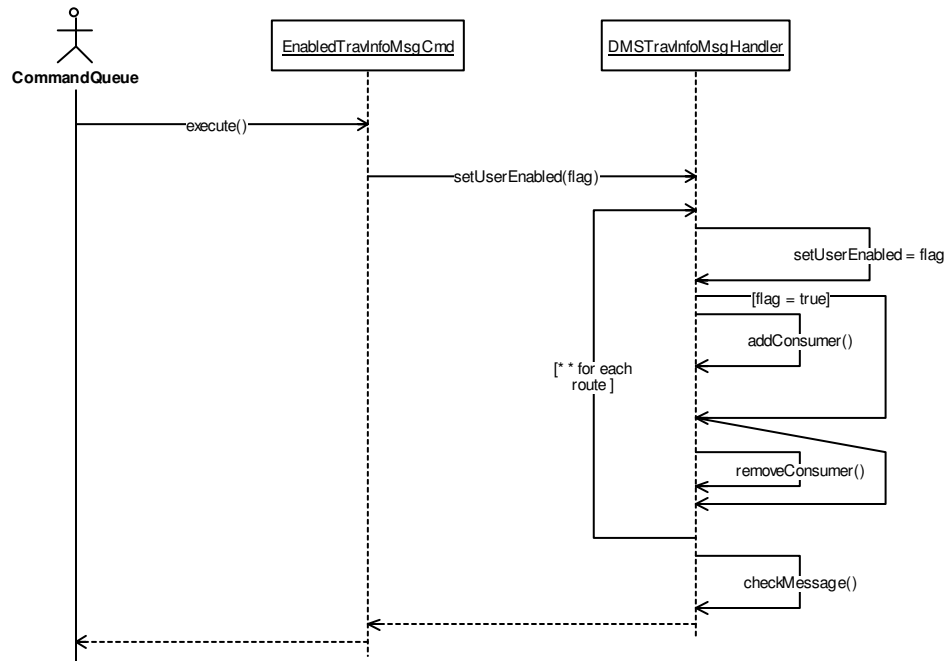


Figure 5-121 EnabledTravInfoMsgCmd:execute (Sequence Diagram)

5.4.2.24 ExternalDMSImpl:GetExternalConfiguration (Sequence Diagram)

This diagram shows the implementation of the GetExternalConfiguratio interface that returns the configuration of external dms.

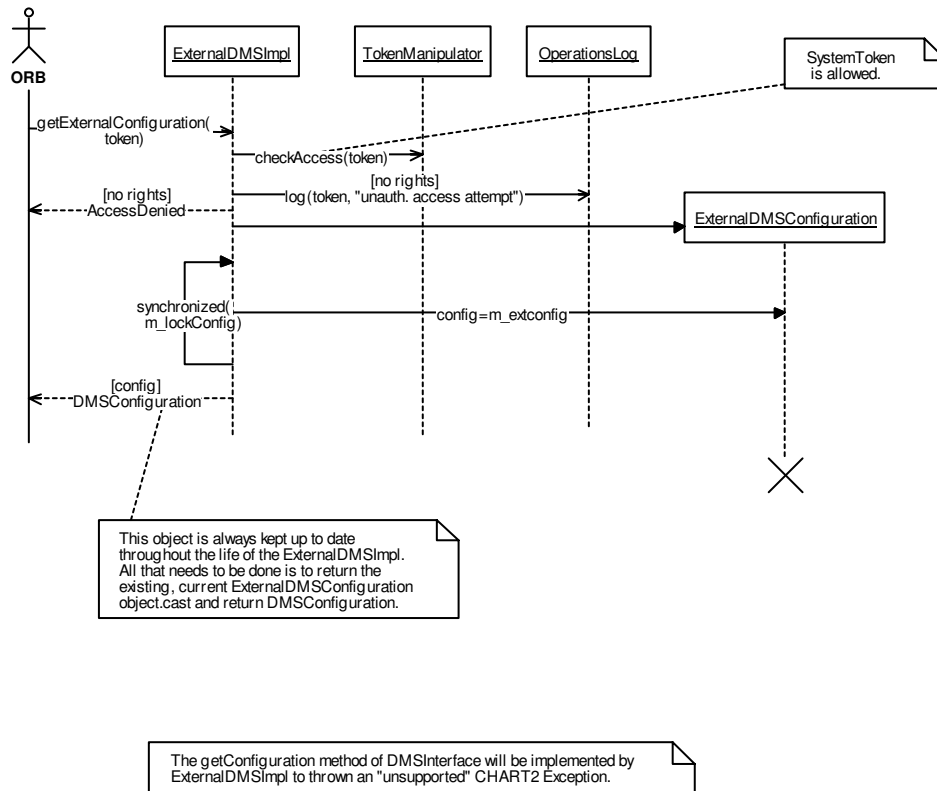


Figure 5-122 ExternalDMSImpl:GetExternalConfiguration (Sequence Diagram)

5.4.2.25 ExternalDMS:GetStatus (Sequence Diagram)

This diagram shows the sequence of getting the status of External DMS from CHART.

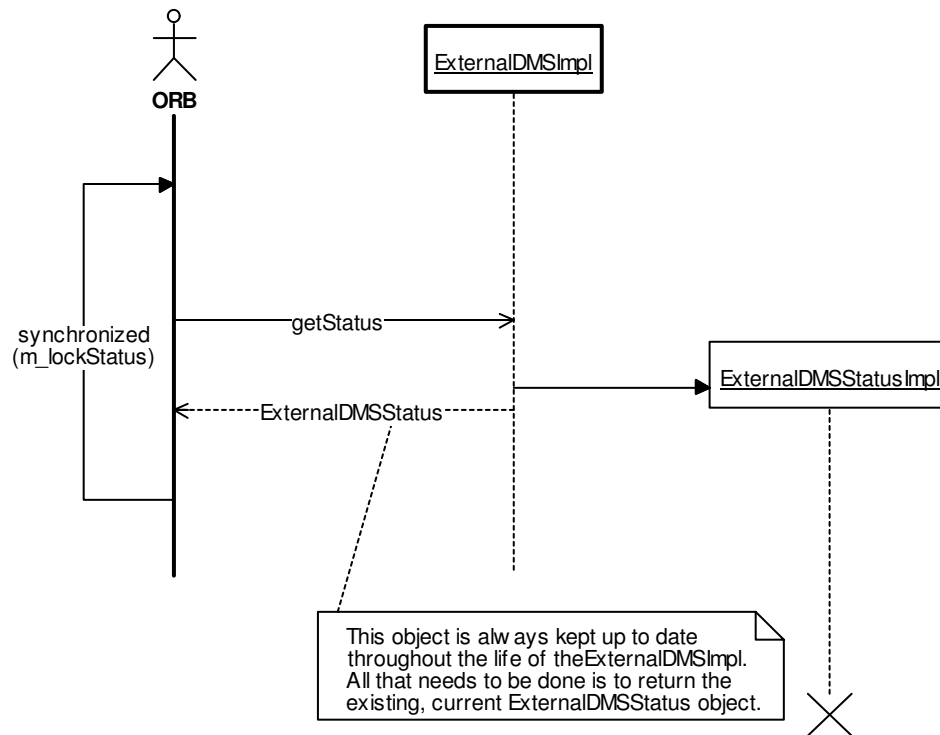


Figure 5-123 ExternalDMS:GetStatus (Sequence Diagram)

5.4.2.26 ExternalDMS:RemoveDMS (Sequence Diagram)

This diagram shows the removal sequence of External DMS in CHART.

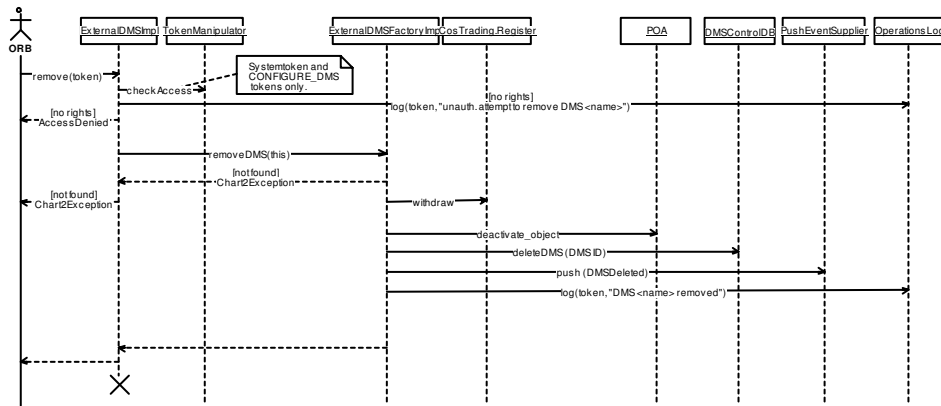


Figure 5-124 ExternalDMS:RemoveDMS (Sequence Diagram)

5.4.2.27 ExternalDMS:SetExternalConfiguration (Sequence Diagram)

This diagram shows the sequence of setting the configuration of External DMS.

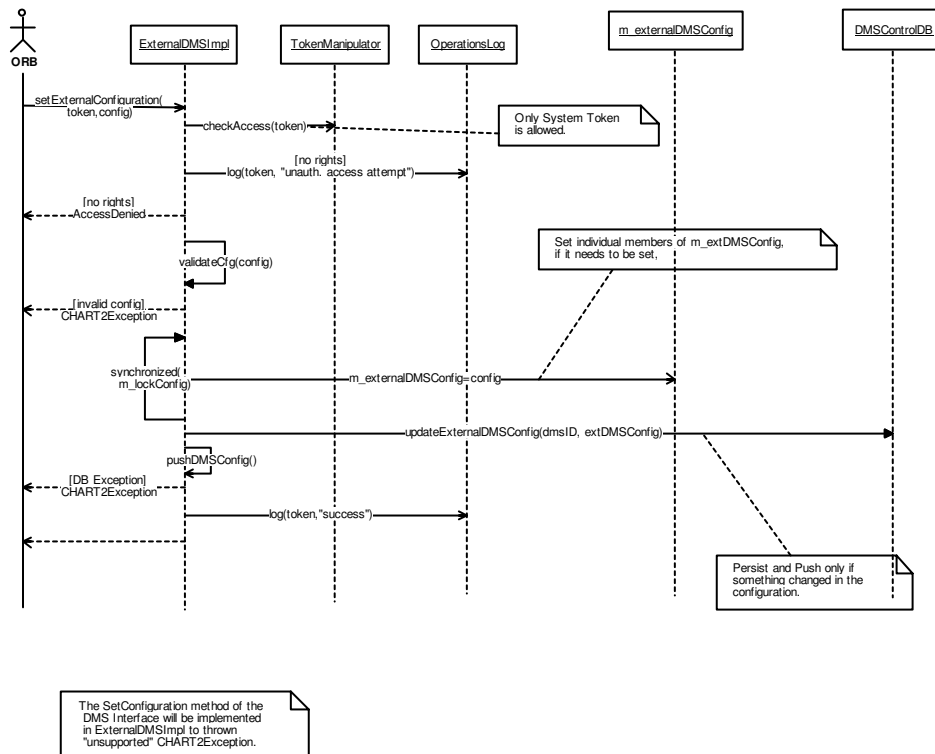


Figure 5-125 ExternalDMS:SetExternalConfiguration (Sequence Diagram)

5.4.2.28 ExternalDMS:updateStatus (Sequence Diagram)

This diagram shows the sequence of updating the status of External DMS when an update is received from the external agency.

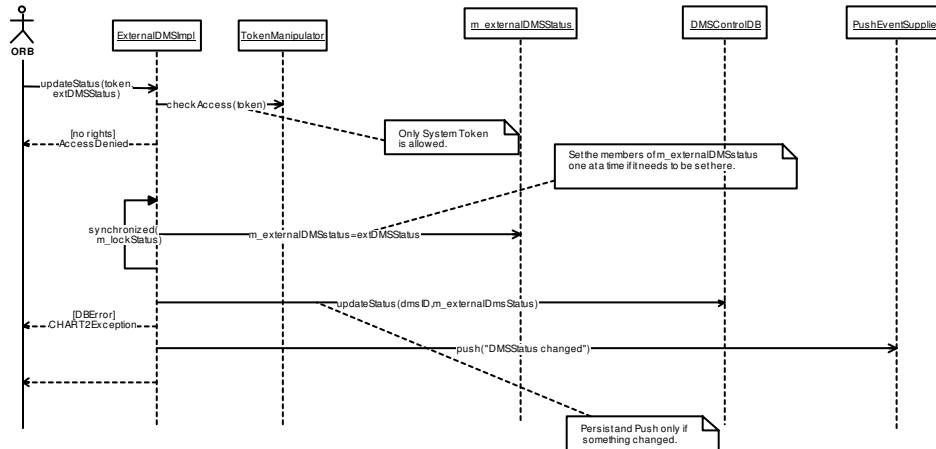


Figure 5-126 ExternalDMS:updateStatus (Sequence Diagram)

5.4.2.29 chartlite.servlet.dms:setDMSConfigCommSettings (Sequence Diagram)

This diagrams shows the processing that occurs when a DMS is configured for TCP/IP communications.

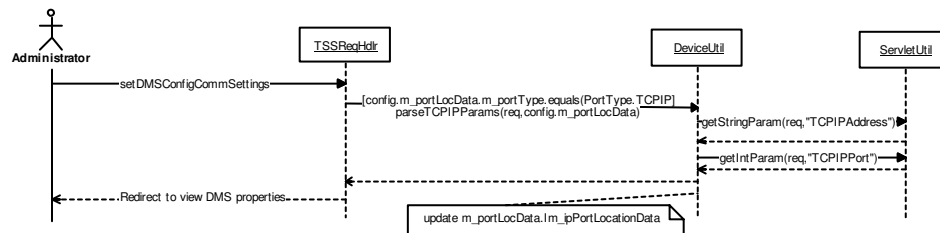


Figure 5-127 chartlite.servlet.dms:setDMSConfigCommSettings (Sequence Diagram)

5.5 DMS Protocols Pkg

5.5.1 Classes

5.5.2 Sequence Diagrams

5.5.2.1 DMSProtocolsPkg:TypicalSetMessage (Sequence Diagram)

This sequence shows typical processing of a protocol handler to set the message of a DMS. All protocol handlers have slightly different implementations due to the different protocols being implemented, however all protocol handlers have a general goal of formatting a byte array according to the device protocol, sending the byte array to the device, and receiving a response from the device to determine if the command was successful. Because DMS messages are specified in the MULTI format, part of the processing required to format a byte array to command the DMS includes converting the MULTI message into the proper sequence of bytes the DMS expects. The MultiConverter class helps to parse through the MULTI tags and pull apart the message into simple pieces that the protocol handler can use to format the byte array. Once told to parse a multi string, the MultiConverter calls back into the parse listener (which happens to be the protocol handler in our case) as it encounters multi tags and message text. After the protocol handler has formatted the byte array, it sends it to the device using the DataPort interface, which may actually be a modem or a direct connect port. After sending the command, the protocol handler reads the response from the device and determines if the command was successful. Failures are indicated through the use of exceptions which contain a specific reason for the failure.

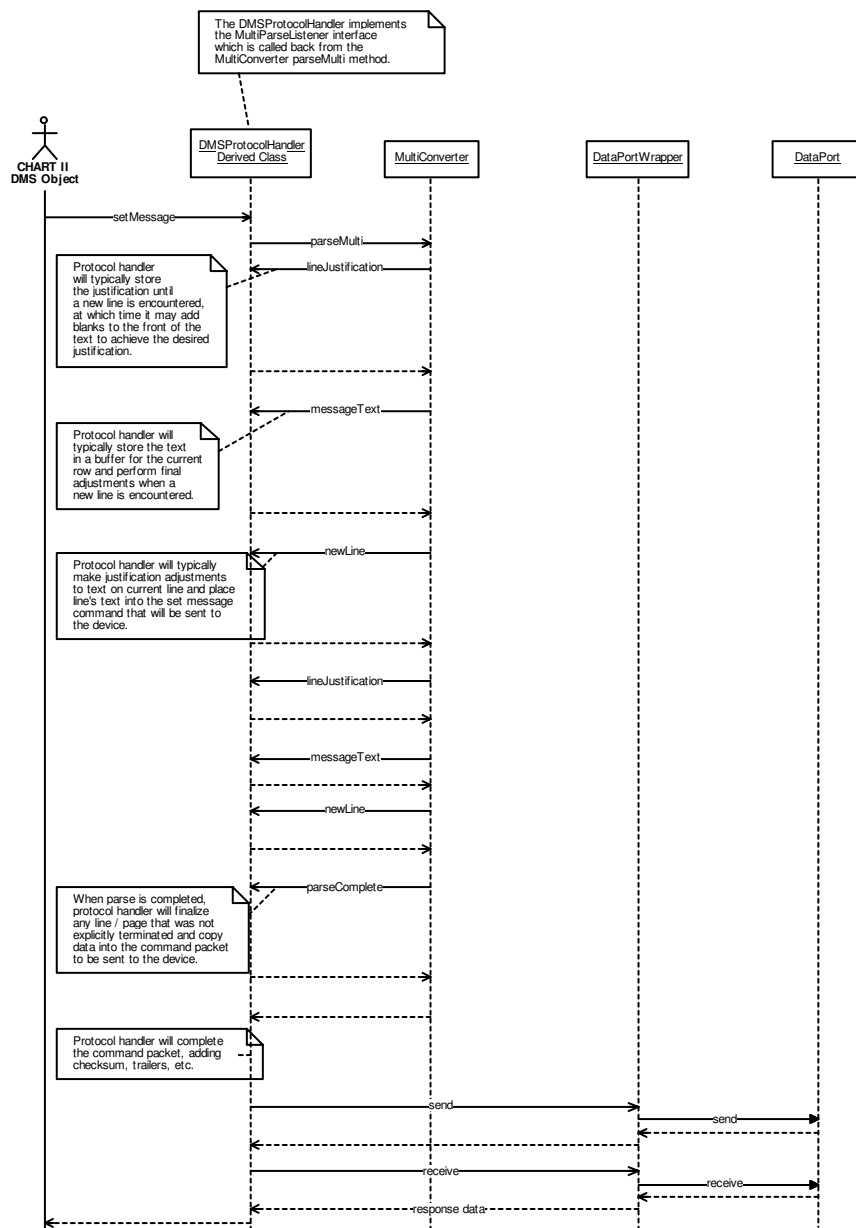


Figure 5-128 DMSProtocolsPkg:TypicalSetMessage (Sequence Diagram)

5.5.2.2 FP9500ProtocolHdlr:GetStatus (Sequence Diagram)

This sequence shows the processing involved in getting the status from the FP9500 device. Since the device updates the pixel status information internally only during a pixel test operation, the caller must have issued a pixel test command prior to get status operation in order to receive the most current status from the device. During get status operation, the Status record is downloaded from the device using the "Parameter Upload" command. Next the lamp status and pixel status bitmaps are downloaded from the device using the "Display Upload" command. If any of the above device commands fail due to response timeout or response format error, a DMSProtocolHandlerException is thrown detailing the failure. On

successful completion of all the above command sequences, the device responses are reformatted and stored in a FP9500DMSDeviceStatus struct and returned to the caller.

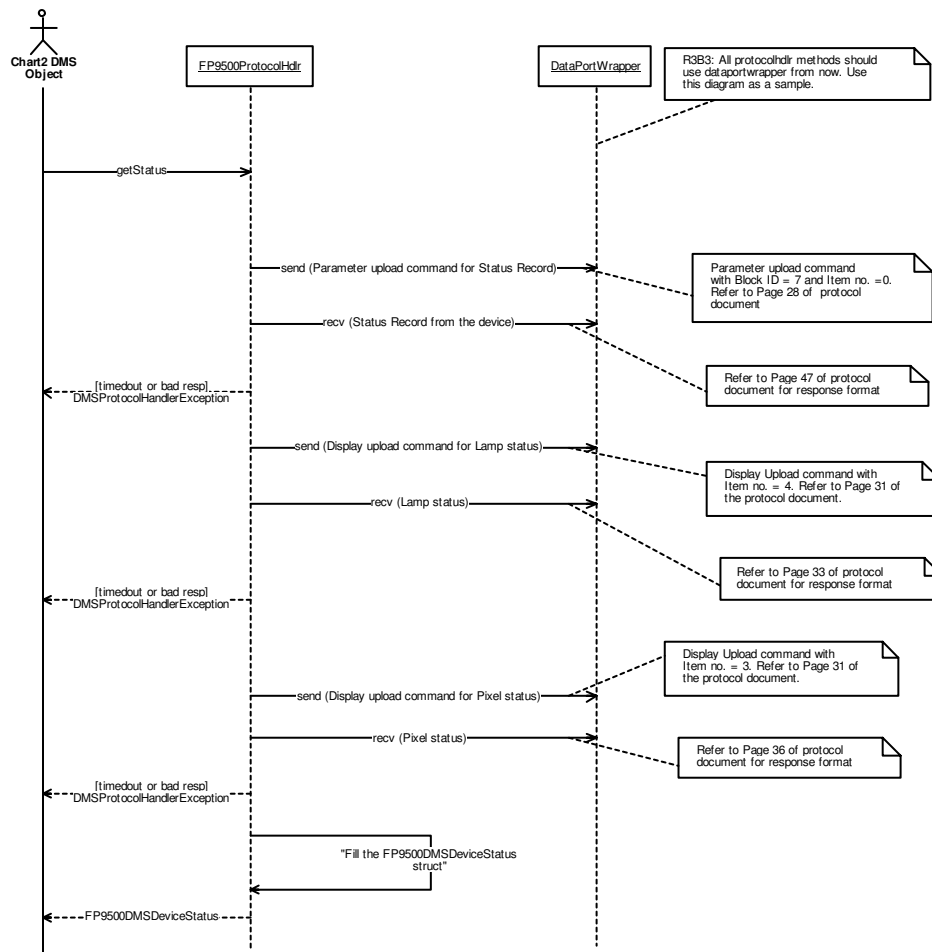


Figure 5-129 FP9500ProtocolHdlr:GetStatus (Sequence Diagram)

5.5.2.3 FP9500ProtocolHdlr:PixelTest (Sequence Diagram)

This sequence shows the processing involved in running a pixel test on the FP9500 device. The FP9500 message selection command with Pixel test option is sent to the device. The response from the device is trivial and indicates the successful start of the pixel test on the sign. The caller may need to allow for a brief interval of time, before any other command is sent to the device. This is to allow the device to run atleast one iteration of the pixel test without interruption and compile the results of the test.

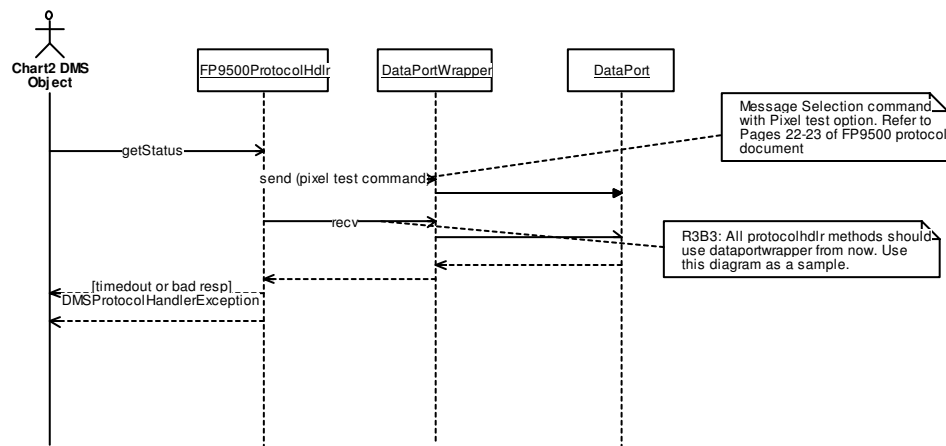


Figure 5-130 FP9500ProtocolHdlr:PixelTest (Sequence Diagram)

5.5.2.4 NTCIPProtocolHdlr:SetMessage (Sequence Diagram)

This sequence diagram shows the steps involved in setting a message on a NTCIP DMS sign.

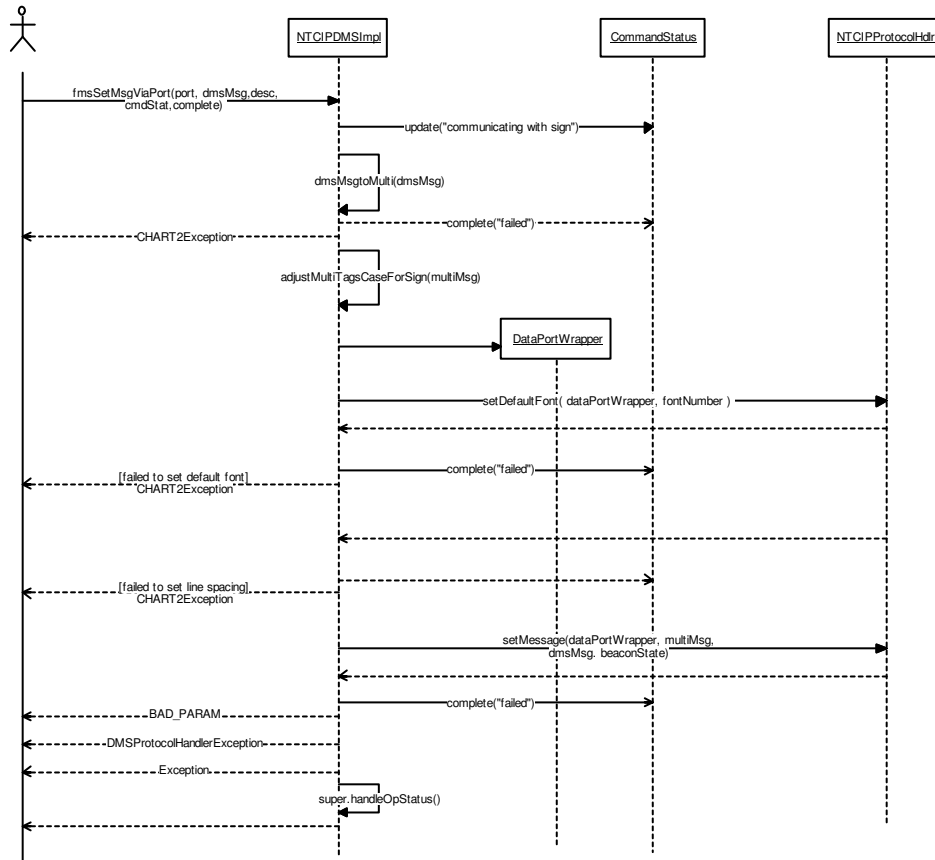


Figure 5-131 NTCIPProtocolHdlr:SetMessage (Sequence Diagram)

5.5.2.5 TS3001ProtocolHdlr:GetStatus (Sequence Diagram)

This sequence shows the processing involved in getting the status from the TS3001 device. First a Sign Status Enquiry command of enquiry type 'S1' is sent to the device. The response to this command contains various sign status information including a brief pixel status and lamp status information. If the response indicates a pixel error, a Sign Status Enquiry command of enquiry type 'S3' is sent to the device. The device responds with a pixel status bitmap. If the 'S1' enquiry response also indicated a lamp error, a Sign Status Enquiry command of enquiry type 'S4' is sent to the device. The device responds with a lamp status bitmap. On successful completion of all the above command sequences, the device responses are reformatted and stored in a TS3001DMSDeviceStatus struct and returned to the caller.

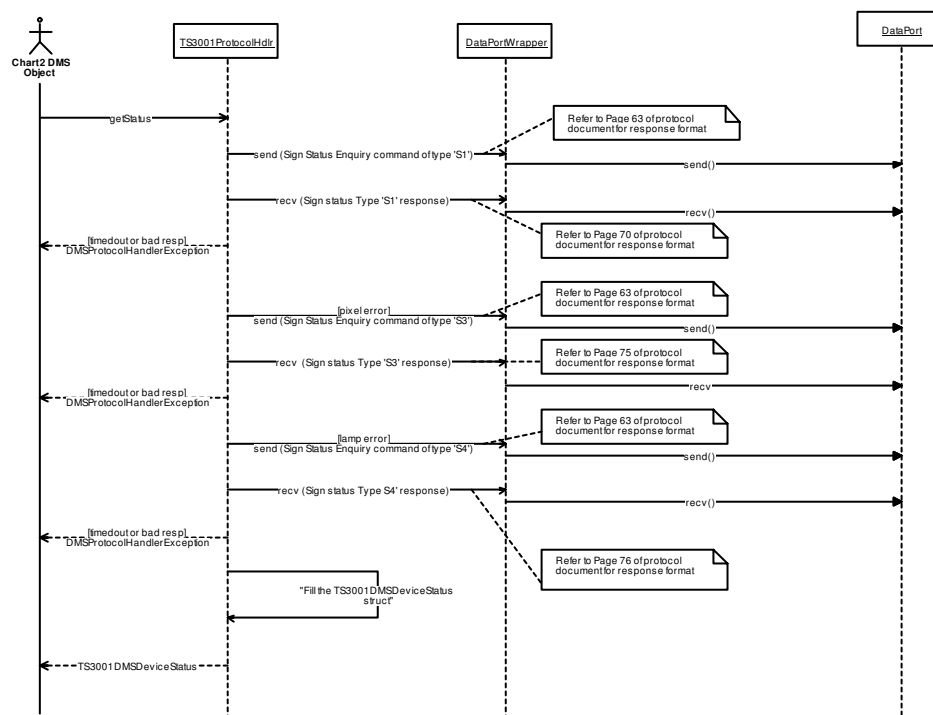


Figure 5-132 TS3001ProtocolHdlr:GetStatus (Sequence Diagram)

5.6 DMSUtilityPkg

5.6.1 Classes

5.6.1.1 DMSUtility (Class Diagram)

This Class Diagram shows classes related to the DMS that are used by both the GUI and the DMS service. Most of these classes are implementations of value type classes defined in the system interfaces (IDL).

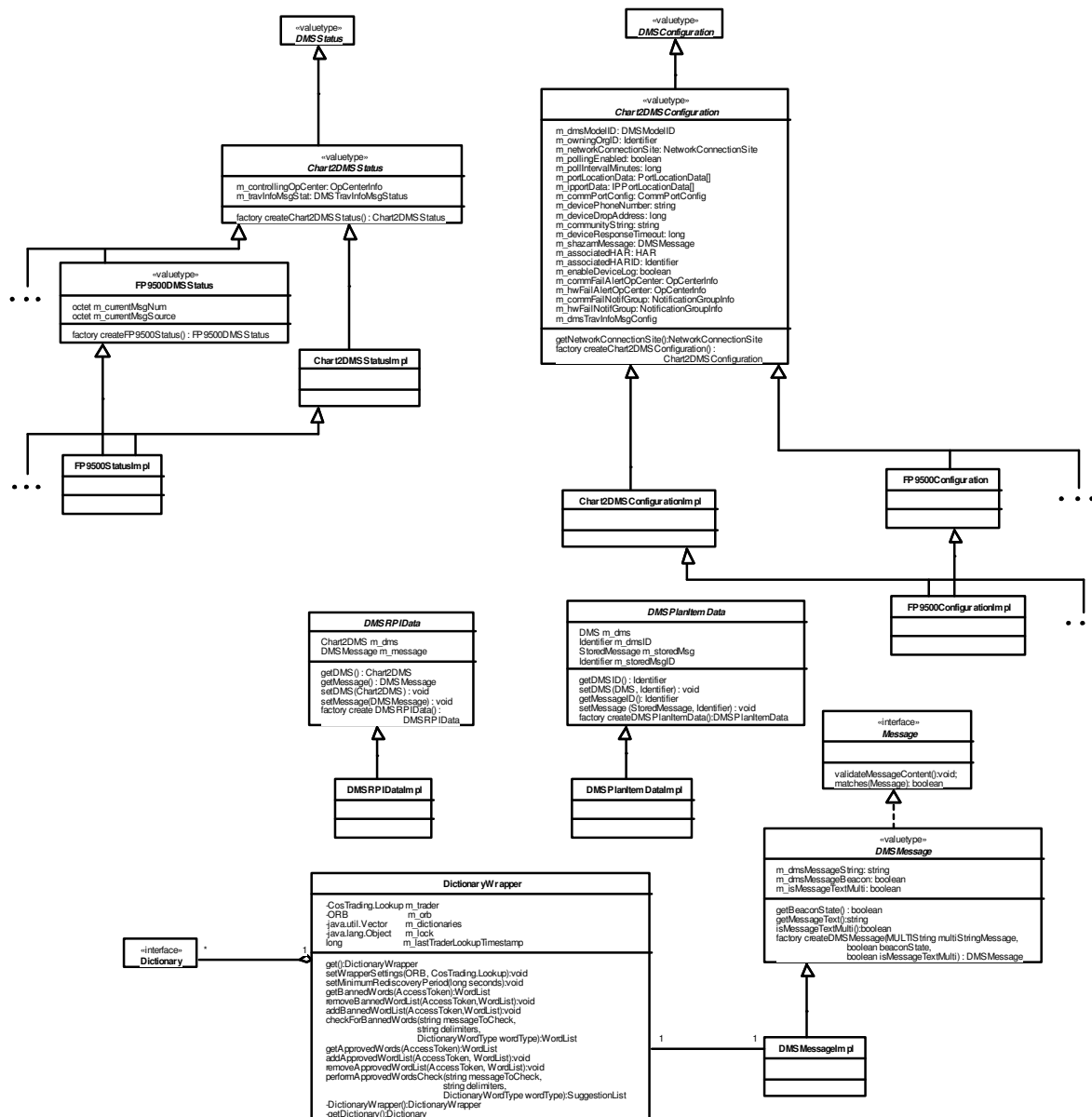


Figure 5-133 DMSUtility (Class Diagram)

5.6.1.1.1 Chart2DMSConfiguration (Class)

The Chart2DMSConfiguration class is an abstract class which extends the DMSConfiguration class to provide configuration information specific to Chart II processing. Such information includes how to contact the sign under Chart II software control, the default SHAZAM message for using the sign as a HAR Notifier, and the owning organization. Such data extends beyond what would be industry-standard configuration information for a DMS. Parameters to support TCP/IP communications, notifications and more alerts, and traveler information messages were added for R3B3.

5.6.1.1.2 Chart2DMSConfigurationImpl (Class)

The Chart2DMSConfigurationImpl class provides an implementation for the abstract Chart2DMSConfiguration class. It implements get and set methods to access and modify values of the configuration of a DMS. The configuration information stored here is normally fairly static: things like the size of the sign in characters and pixels, its name and location, and how to contact the sign (as opposed to dynamic information like the current message on the sign, which is stored in an analogous Status object).

5.6.1.1.3 Chart2DMSStatus (Class)

The Chart2DMSStatus class is an abstract class which extends the DMSSStatus class to provide status information specific to CHART processing, such as information on the controlling operations center for the sign. This data extends beyond what would be industry-standard status information for a DMS. Status information for traveler information messages was added in R3B3.

5.6.1.1.4 Chart2DMSStatusImpl (Class)

The Chart2DMSStatusImpl class provides an implementation for the abstract Chart2DMSStatus class. It implements get and set methods to access and modify values of the status of a DMS. The status information stored here is relatively dynamic: things like the current message on the sign, its beacon state, its current operational mode (online, offline, maintenance mode), and current operational status (OK, COMM_FAILURE, or HARDWARE_FAILURE) and controlling operations center. (More static information about the sign, such as its size and location, is stored in an analogous Configuration object.)

5.6.1.1.5 Dictionary (Class)

The Dictionary IDL interface provides functionality to add, delete and check for words that are approved or banned from being used in CHART2 messaging devices such as HARs and DMSs. It also provides functionality to manage pronunciations.

5.6.1.1.6 DictionaryWrapper (Class)

This singleton class provides a wrapper for the system dictionary that provides automatic location of the dictionary and automatic re-discovery should the dictionary reference return an error. This class also allows for built-in fault tolerance by automatically failing over to a "working" dictionary without the user of this class being aware that this being done. In addition, this class defers the discovery of the Dictionary until its first use, thus eliminating a start-up dependency for modules that use the dictionary.

This class delegates all of its method calls to the system dictionary using its currently known good reference to the system dictionary. If the current reference returns a CORBA failure in the delegated call, this class automatically switches to another reference. When there are no good references (as is true the first time the object is used), this class issues a trader query to (re)discover the published Dictionary objects in the system. During a method call, the trader will be queried at most one time and under normal circumstances

(other than the first use) the trader will not be queried at all.

5.6.1.1.7 DMSConfiguration (Class)

The DMSConfiguration class is an abstract valuetype class which describes the configuration of a DMS device. This configuration information is normally fairly static: things like the size of the sign in characters and pixels, its name and location, and how to contact the sign (as opposed to dynamic information like the current message on the sign, which is defined in an analogous Status object). The font number and line spacing were added for R3B3, and the location was changed to a ObjectLocation, which contains more detailed locations fields.

5.6.1.1.8 DMSMessage (Class)

The DMSMessage class is an abstract class which describes a message for a DMS. It consists of two elements: a MULTI-formatted message and beacon state information (whether the message requires that the beacons be on). The DMSMessage is contained within a DMSStatus object, used to communicate the current message on a sign, and is stored within a DMSRPIData object, used to specify the message which should be on a sign when the response plan item is executed.

5.6.1.1.9 DMSMessageImpl (Class)

The DMSMessageImpl class provides an implementation for the abstract DMSMessage class. It implements get and set methods to access and modify the MULTI-formatted message and beacon state values which make up a DMS message.

5.6.1.1.10 DMSPlanItemData (Class)

The DMSPlanItemData class is a valuetype that contains data stored in a plan item for a DMS. It is derived from PlanItemData.

5.6.1.1.11 DMSPlanItemDataImpl (Class)

The DMSPlanItemDataImpl class provides an implementation for the abstract DMSPlanItemData class. It implements get and set methods to access and modify values relative to a stored Plan Item for a DMS, which associates a stored message to a specific DMS it should be placed on.

5.6.1.1.12 DMSRPIData (Class)

The DMSRPIData class is an abstract class which describes a response plan item for a DMS. It contains the unique identifier of the DMS to contain the DMSMessage, and the DMSMessage itself.

5.6.1.1.13 DMSRPIDataImpl (Class)

The DMSRPIDataImpl class provides an implementation for the abstract DMSRPIData class. It implements get and set methods to access and modify values relative to a

Response Plan Item for a DMS.

5.6.1.1.14 DMSStatus (Class)

The DMSStatus class is an abstract value-type class which provides status information for a DMS. This status information is relatively dynamic: things like the current message on the sign, its beacon state, its current operational mode (online, offline, maintenance mode), and current operational status (OK, COMM_FAILURE, or HARDWARE_FAILURE). (More static information about the sign, such as its size and location, is defined in an analogous Configuration object.)

5.6.1.1.15 FP9500Configuration (Class)

The FP9500Configuration class is an abstract class which extends the Chart2DMSConfiguration class to provide configuration information specific to an FP9500 model of DMS. It is exemplary of potentially a whole suite of subclasses specific to a specific brand and model of sign for manufacturer-specific configuration information.

5.6.1.1.16 FP9500ConfigurationImpl (Class)

The FP9500ConfigurationImpl class provides an implementation for the abstract FP9500Configuration class. It implements get and set methods to access and modify values specific to the static configuration of an FP9500 DMS. It is exemplary of potentially a whole suite of subclasses specific to a specific brand and model of sign for manufacturer-specific configuration information.

5.6.1.1.17 FP9500DMSStatus (Class)

The FP9500DMSStatus class provides additional storage for status information unique to the FP9500 model of sign. It is exemplary of potentially a whole suite of Chart2DMSStatus subclasses specific to a specific brand and model of sign.

5.6.1.1.18 FP9500StatusImpl (Class)

The FP9500StatusImpl class provides an implementation for the abstract FP9500Status class. It implements get and set methods to access and modify values specific to the dynamic status configuration of an FP9500 DMS. It is exemplary of potentially a whole suite of subclasses specific to a specific brand and model of sign for manufacturer-specific status information.

5.6.1.1.19 Message (Class)

This class represents a message that will be used while activating devices. This class provides a means to check if the message contains any banned words given a Dictionary object. Derived classes extend this class to provide device specific message data.

5.6.1.2 DMSTravInfoMsgFormattingClasses (Class Diagram)

This diagram contains classes used in formatting the MULTI for a DMSTravInfoMsgTemplate.

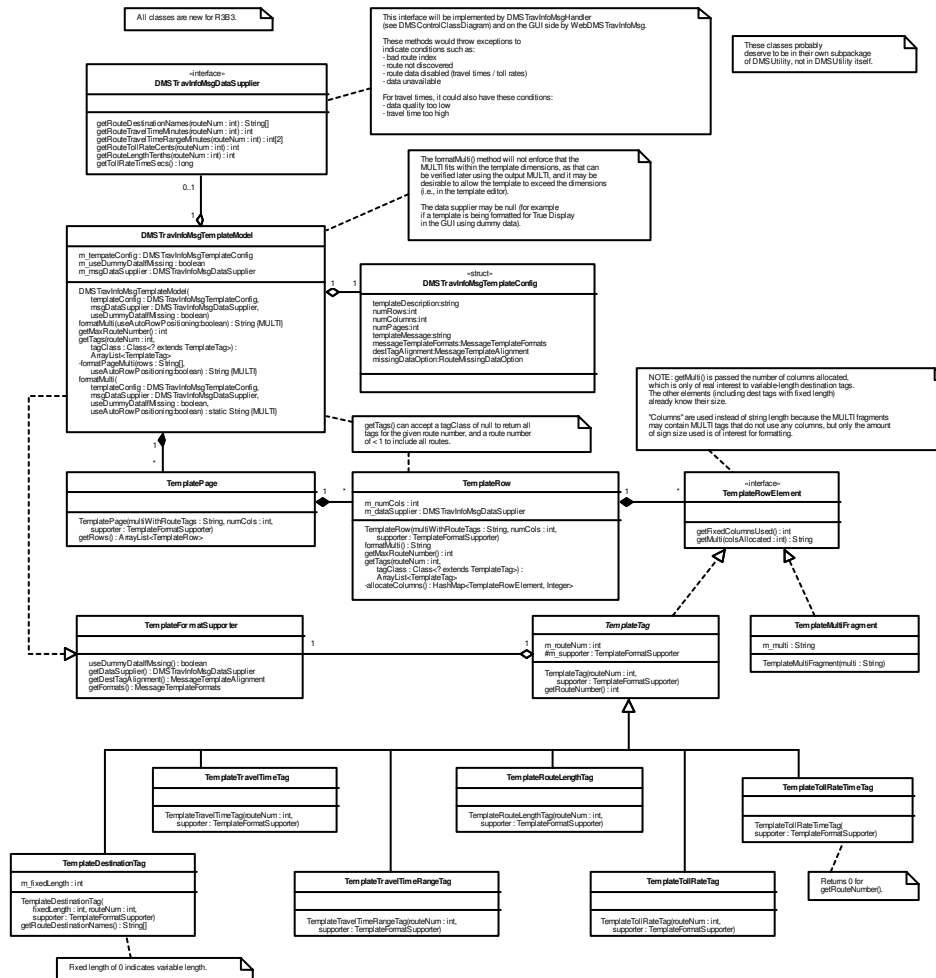


Figure 5-134 DMSTravInfoMsgFormattingClasses (Class Diagram)

5.6.1.2.1 DMSTravInfoMsgDataSupplier (Class)

This interface provides data for travel routes used in a DMSTravInfoMsg. It will be used to substitute the template tags with route-specific data, in order to format the template and produce MULTI. This is needed in the GUI for true display, and is needed in the server for formatting messages to send to a DMS. The routeNum parameter corresponds to route numbers contained in the template data tags, and it is a 1-based index. These methods will throw an exception if the requested data is not available.

5.6.1.2.2 DMSTravInfoMsgTemplateConfig (Class)

This object contains the configuration data for a message template that represents a

DMSTravlInfoMsgTemplate in the CHART DB

5.6.1.2.3 DMSTravInfoMsgTemplateModel (Class)

This class contains functionality for formatting and modelling DMS message templates. During initialization a model of pages, rows, and elements (including the template tags) is constructed. MULTI fragments (the MULTI outside of the template tags) are stored so that they can be carried to the formatted MULTI. The tags can also be queried from the model, which can be used to figure out what data will be required for each route by the template.

5.6.1.2.4 TemplateDestinationTag (Class)

This class represents a route destination data tag within the template. The destination field can have a fixed or variable size, and the length will be set to 0 to indicate that the size is variable (if applicable).

5.6.1.2.5 TemplateFormatSupporter (Class)

This interface provides the tag classes with the data they need to format the route data for the tag into MULTI.

5.6.1.2.6 TemplateMultiFragment (Class)

This class is the portion of a template row between the data tags (if applicable). The MULTI is kept intact so that MULTI tags such as line justification can be preserved in the output MULTI.

5.6.1.2.7 TemplatePage (Class)

This class represents a page of the template that has been parsed into model form. It contains TemplateRow objects which represent the rows of the page.

5.6.1.2.8 TemplateRouteLengthTag (Class)

This class represents a route length tag within the template.

5.6.1.2.9 TemplateRow (Class)

This class represents a row of the template page that has been parsed into model form. It contains TemplateElement objects which represent the data tags and non-data tag MULTI fragments of the row.

5.6.1.2.10 TemplateRowElement (Class)

This interface represents an element (component) of the message template row, which can either be a data tag or a fragment of the MULTI outside of the tags.

5.6.1.2.11 TemplateTag (Class)

This is an abstract base class for template tags. It contains the route number for the tag,

which is a number specified in the template tag and is greater than or equal to one. For a `TemplateTollRateTimeTag` however, the route number will be zero to indicate that it is not tied to a specific route.

5.6.1.2.12 `TemplateTollRateTag` (Class)

This class represents a route toll rate tag within the template.

5.6.1.2.13 `TemplateTollRateTimeTag` (Class)

This class represents a toll rate time tag within the template.

5.6.1.2.14 `TemplateTravelTimeRangeTag` (Class)

This class represents a route travel time range tag within the template.

5.6.1.2.15 `TemplateTravelTimeTag` (Class)

This class represents a route travel time tag within the template.

5.6.2 Sequence diagrams

5.6.2.1 DMSTravInfoMsgTemplateModel:formatMulti (Sequence Diagram)

This diagram shows how the MULTI message is formatted using the DMSTravInfoMsgTemplateFormatter, which has already been set up at construction time to model the template's pages and rows. Each TemplatePage object is called to get its TemplateRow objects. The row is called to format its own MULTI, which includes formatting any data tags (as shown in the TemplateRow:FormatMulti sequence diagram). If the returned MULTI is null, an error occurred getting the route data and depending on the missing data option in the template, if the message is to be ignored, it will return an empty string; otherwise, if the page is to be ignored, it will skip to the next page and avoid generating MULTI for the current page. Otherwise, if non-empty MULTI is returned, it is stored in the array of row MULTI strings. The array may contain null values for unused rows. Next the formatPageMulti() is called to get the MULTI for the page, applying the automatic row positioning flag if specified. If it's not the first page, the "new page" tag is added before the page multi is added. Finally the MULTI is returned.

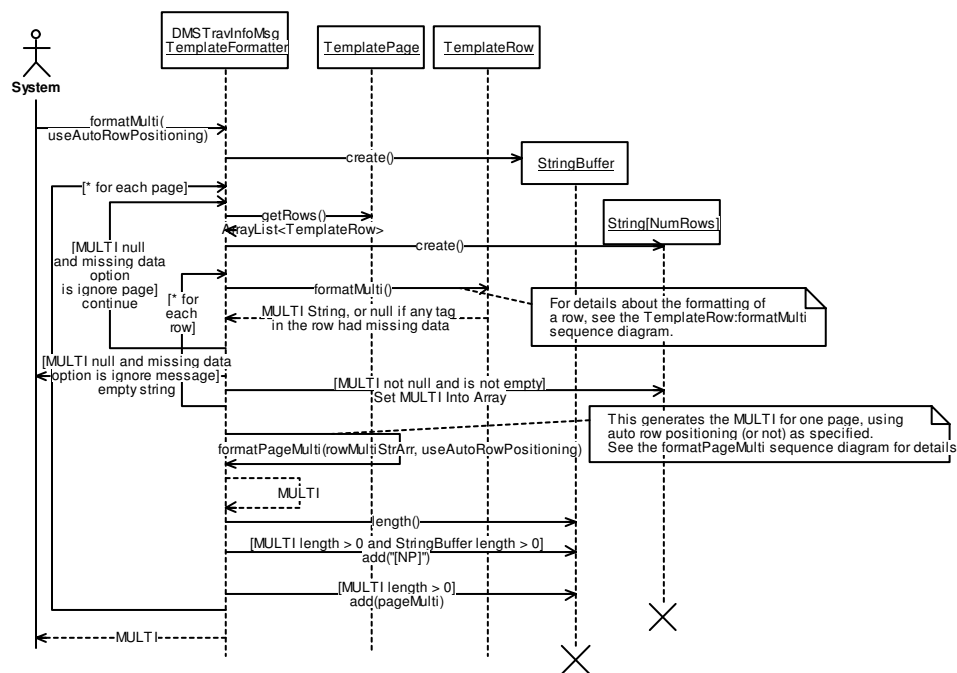


Figure 5-135 DMSTravInfoMsgTemplateModel:formatMulti (Sequence Diagram)

5.6.2.2 DMSTravInfoMsgTemplateModel:formatPageMulti (Sequence Diagram)

This diagram shows how the row MULTI strings are formatted to obtain the MULTI for a

DMS page. If the "useAutoRowPositioning" flag is true, a new ArrayList is created and any non-null elements of the input row string array are added to the list. A new row string array is created, and the rows are positioned within the array corresponding to the auto positioning rules in the requirements. The result of autopositioning is that the input array has been replaced by a new array, with the rows in their new positions. The elements of the array are then examined to produce the output MULTI. If the array element is not null, it is appended to the output MULTI. Regardless of whether the element was null or not, a newline ("NL") tag is appended to the MULTI unless it is the last line on the page (i.e., the last element in the array). The MULTI is returned to the caller.

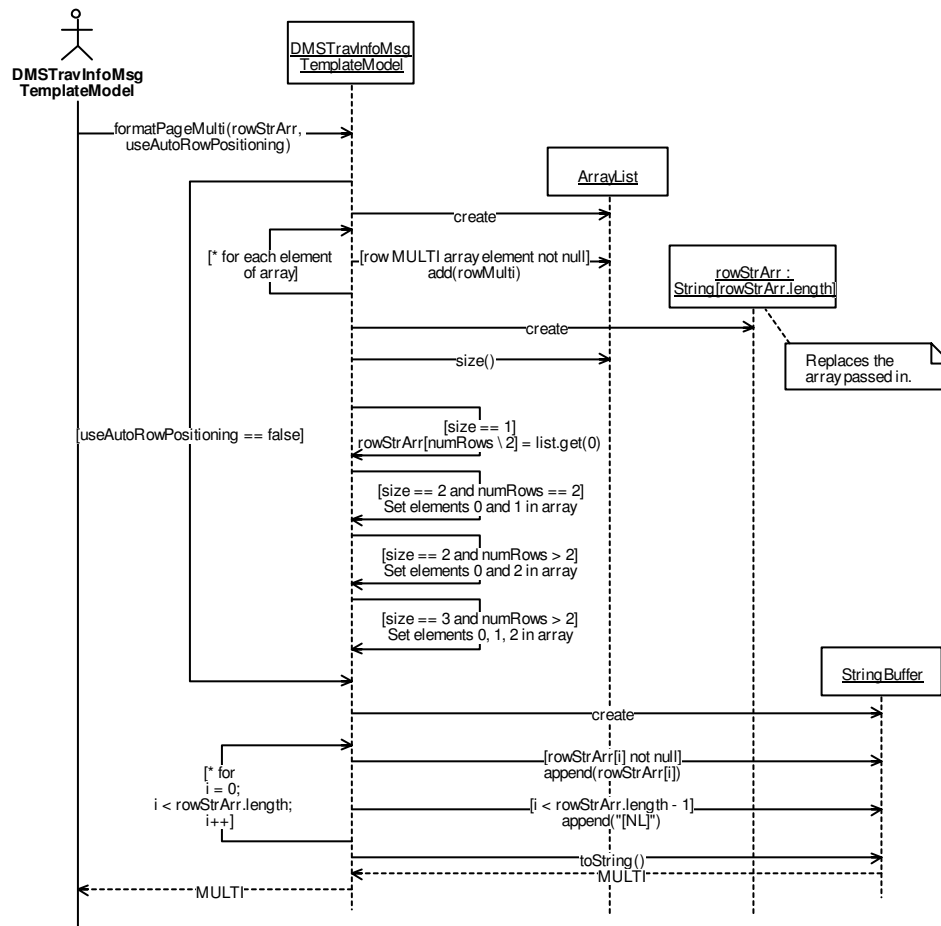


Figure 5-136 DMSTravInfoMsgTemplateModel:formatPageMulti (Sequence Diagram)

5.6.2.3 TemplateRow:formatMulti (Sequence Diagram)

This diagram shows how the MULTI for a row of the template is built. Each row element is called to get its fixed columns used. This will return a positive number except if it is a variable length tag (i.e., a destination tag that is not fixed width). If the positive value is found, it's added to the HashMap for the tag, and it is added to the total for fix columns

[illegible]

12/23/2008

5.7 DeviceUtilityPkg

5.7.1 Classes

5.7.1.1 PortLocatorClasses (Class Diagram)

This class diagram shows utility classes that can be used to get a free port.

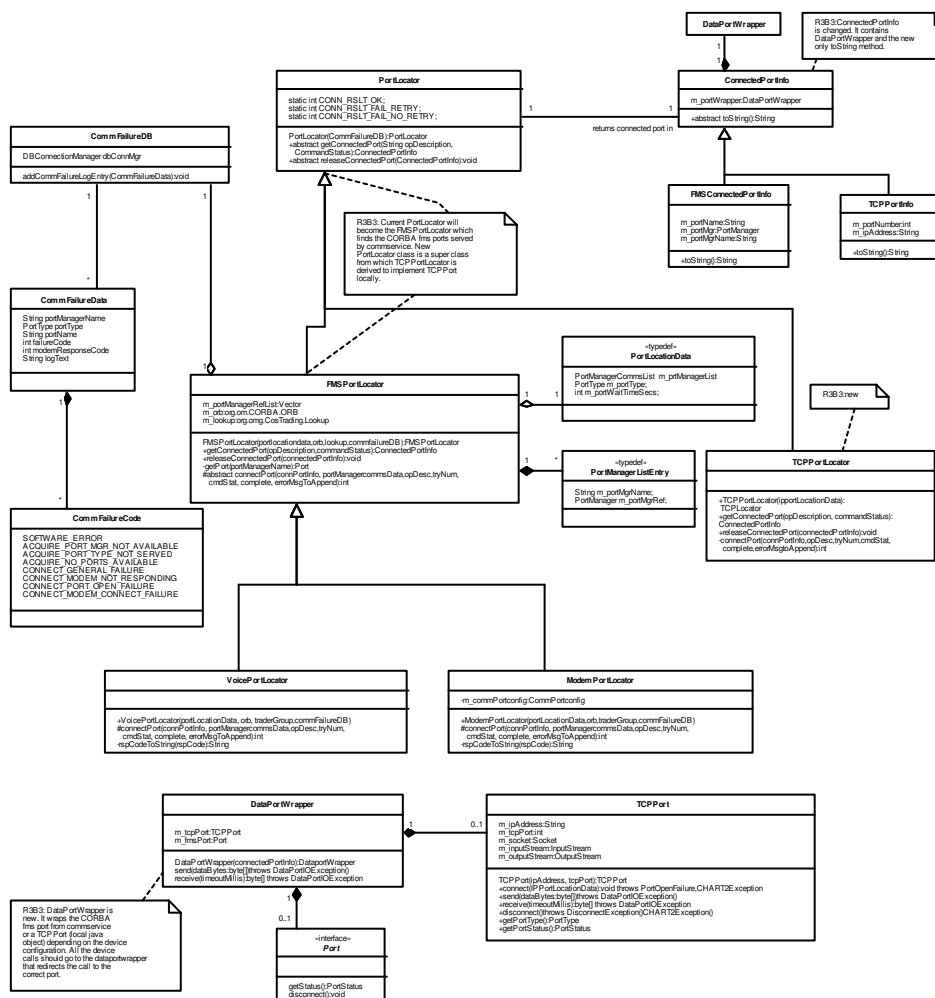


Figure 5-138 . PortLocatorClasses (Class Diagram)

5.7.1.1.1 CommFailureCode (Class)

This class defines static values to be used to specify the type of comm failure in a CommFailureData object.

5.7.1.1.2 CommFailureData (Class)

This class holds data to be passed to the CommFailureDB class to be logged in the Comm failure log in the database.

5.7.1.1.3 CommFailureDB (Class)

This class is a utility used to log an entry in the Comm Failure log table in the database. This table is used to log details about any comm failure that occurs in the system.

5.7.1.1.4 ConnectedPortInfo (Class)

This class holds data pertaining to a port that was acquired and connected via the PortLocator.

5.7.1.1.5 DataPortWrapper (Class)

This class is a wrapper for FMS port objects and TCPPort objects . All device port communications will be routed through the DataPortWrapper to the correct object depending on the device configuration.

5.7.1.1.6 FMSCONNECTEDPortInfo (Class)

This class contains the information about the fms port connected through the FMS portmanager

5.7.1.1.7 FMSPortLocator (Class)

The FMSPortLocator is a utility class that helps one to utilize the fault tolerance provided by the deployment of many PortManagers. The FMSPortLocator is initialized by specifying a preferred PortManager and optionally one or more alternate PortManagers using a PortLocationData object.

When asked to get a connected port, the PortLocator first attempts to acquire a port from the preferred PortManager and then calls its abstract connectPort() method (implemented by derived classes) to attempt to connect to the port. If a failure occurs, the FMSPortLocator retries the sequence using the next PortManager in the list. The list may contain the same port manager multiple times to have retries occur on the same port manager prior to moving to another. In the event that the FMSPortLocator will perform a retry on the same port manager, it holds the previously acquired port while performing the retry to avoid having the port manager return the same port during the retry. When a different port is acquired during a retry on the same port manager, the port is released (prior to connecting the 2nd port).

5.7.1.1.8 ModemPortLocator (Class)

This class implements the methods to enable tcp/ip port communication of devices.

5.7.1.1.9 Port (Class)

A Port is an object that models a physical communications resource. Derived interfaces specify various types of ports. All ports must be able to supply their status when requested.

5.7.1.1.10 PortLocationData (Class)

This class contains configuration data that specifies the communication server(s) to use to communicate with a device.

m_commsData - One or more objects identifying the communications server (PortManager) to use to communicate with the device, in order of preference.

m_portType - The type of port to use to communicate with the device (ISDN modem, POTS modem, direct, etc.)

m_portWaitTimeSecs - The maximum number of seconds to wait when attempting to acquire a port from a port manager.

5.7.1.1.11 PortLocator (Class)

The PortLocator is a utility class that helps one to connect to the port used by the device. The actual implementation of the operations is done by the derived classes depending on what protocol is used for communication.

5.7.1.1.12 PortManagerListEntry (Class)

This class is used by the PortLocator to map object identifiers to object references for PortManager objects.

5.7.1.1.13 TCPPort (Class)

This class implements the methods to enable tcp/ip port communication of devices.

5.7.1.1.14 TCPPortInfo (Class)

This structure has the information required to enable tcp/ip port communication of devices

5.7.1.1.15 TCPPortLocator (Class)

TCPPortLocator is a utility class that helps to establish and manage connection to a tcpip port.

5.7.1.1.16 VoicePortLocator (Class)

This class provides an implementation of the PortLocator's abstract connectPort() method that can connect a VoicePort that has been acquired by the PortLocator base class. This derived class logs information in the comm failure database table relating to connection problems that may occur. Since this is a telephony port which is much simpler to connect than, say, a ModemPort, there will be considerably fewer types of errors which can occur

and thus be detected and reported.

5.7.2 Sequence Diagrams

5.7.2.1 PortLocator:ReleasePort (Sequence Diagram)

When the PortLocator releasePort method is called, the PortLocator uses the port manager reference that it stored in the getPort method to release the port from the correct PortManager.

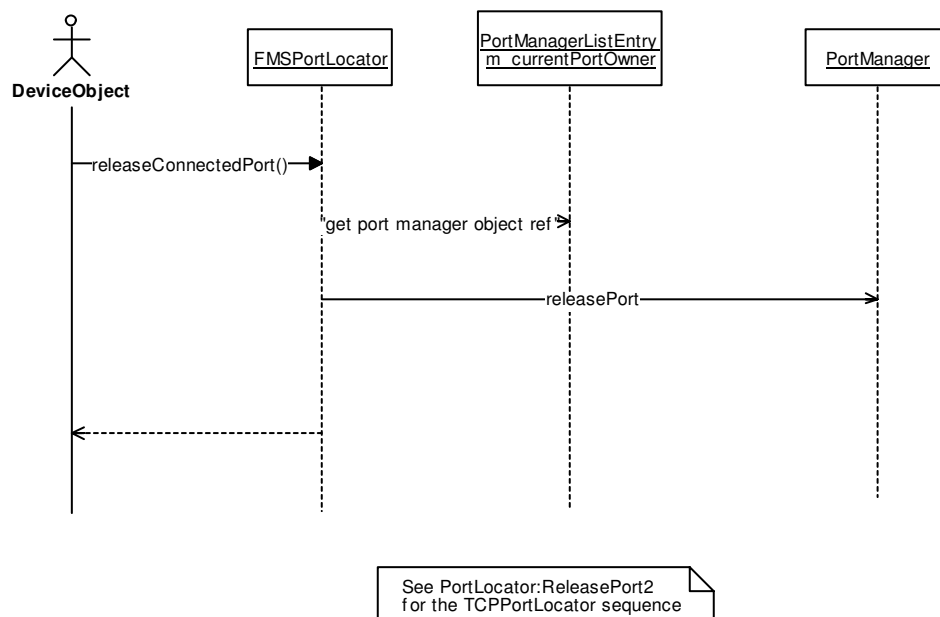


Figure 5-139 PortLocator:ReleasePort (Sequence Diagram)

5.7.2.2 PortLocator:ReleasePort2 (Sequence Diagram)

The TCPPortLocator deletes the TCPPort.

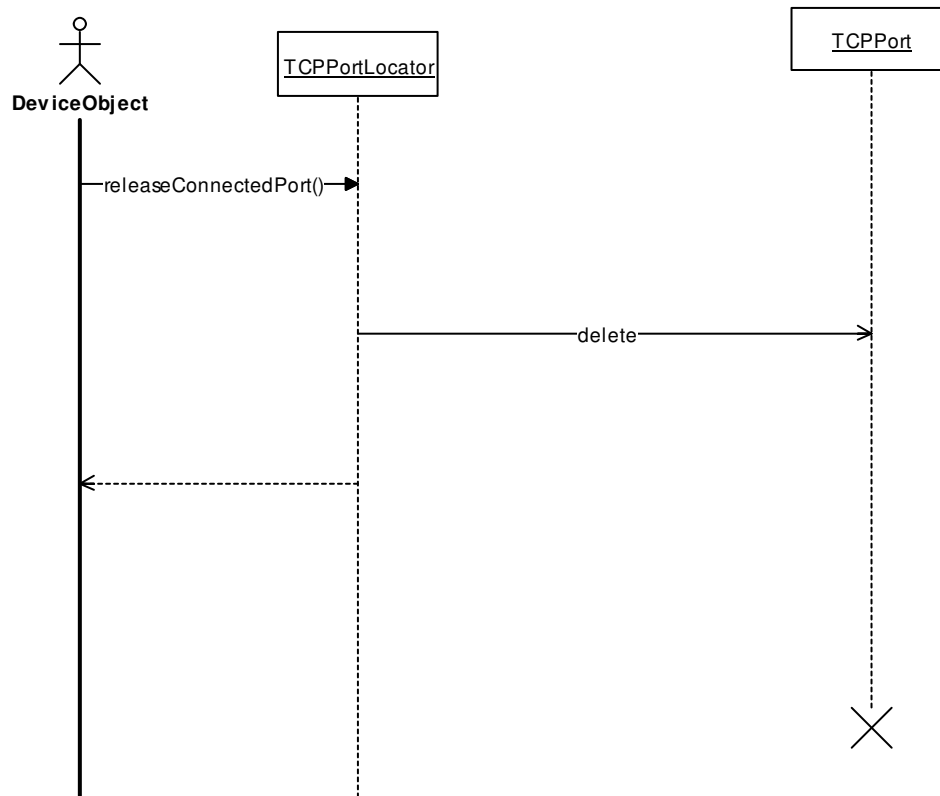


Figure 5-140 PortLocator:ReleasePort2 (Sequence Diagram)

5.7.2.3 PortLocator:getConnectedPort (Sequence Diagram)

The getConnectedPort method of the PortLocator utility uses the list of PortManager names and associated connection information (such as phone number to use) to attempt to acquire a port and connect it to the remote destination. Retry logic exists to try each PortManager in succession until a port is successfully connected or an attempt to connect fails and the type of failure is not likely to benefit from a retry on a different port. The list of port manager names can contain duplicate entries to cause the port locator to use a different port on the same port manager. When this is the case, the port locator must hold the previously acquired port while it attempts to get an additional port from the port manager to ensure the port manager doesn't return the same port twice.

The connection logic is carried out in the derived class connectPort() method, for this logic varies depending on the type of port requested. A private getPort() method handles logic to retrieve a port from a single port manager and process errors. Sequences for these methods exist in the ModemPortLocator:connectPort() sequence and the PortLocator:getPort() sequence.

If the connection attempt succeeds, the device object creates a DataPortWrapper object which points to the DataPort (CORBA object served by the fms) . If the connection attempt fails, appropriate error record is inserted in the CommFailureDB.

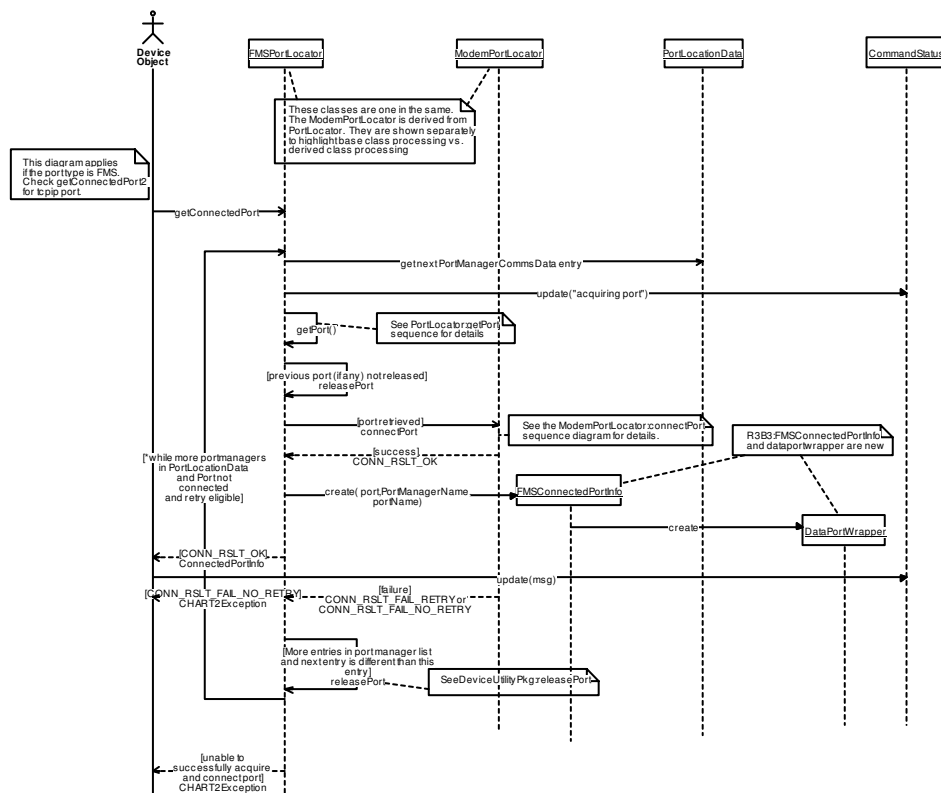


Figure 5-141 PortLocator::getConnectedPort (Sequence Diagram)

5.7.2.4 PortLocator:getConnectedPort2 (Sequence Diagram)

The getConnectedPort method of the PortLocator utility gets the port type and if it is a tcp port it attempts to acquire a port and connect it to the remote destination. If the connection attempt succeeds, the device object creates a DataPortWrapper which points to the newly created and connected port.

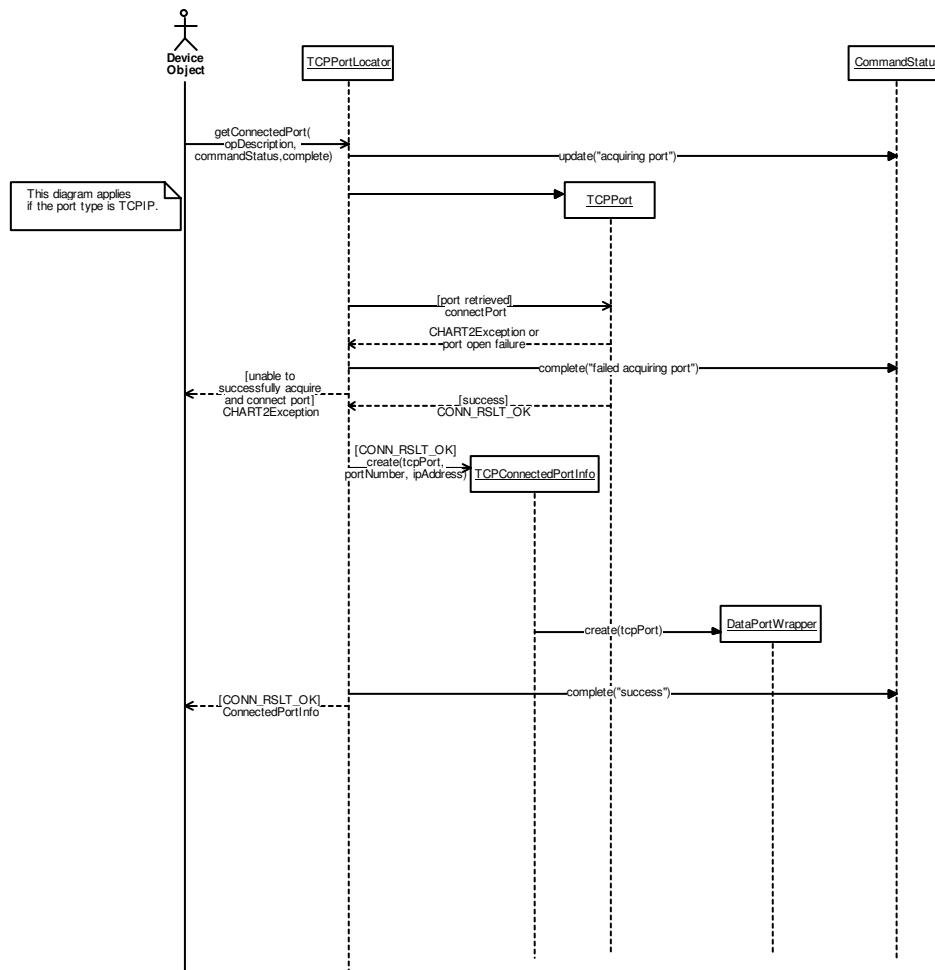


Figure 5-142 PortLocator:getConnectedPort2 (Sequence Diagram)

5.7.2.5 PortLocator:getPort (Sequence Diagram)

When a request is made to the PortLocator to get a port, the PortLocator gets the first entry from its port manager list (which is the preferred port manager) and asks the port manager for a port. If the getPort call on the port manager fails, the PortLocator consults its retry settings and may retry the operation depending on the specific failure condition. If all retries (if any) of the getPort operation on the port manager are exhausted without success,

the PortLocator may failover to the next PortManager in the PortLocator's list, depending on the specific error condition encountered and the settings for failover in the PortLocator. Because the PortLocator is initialized with only object identifiers for the preferred and fallback port managers, a Trader query is made to obtain an object reference the first time a PortManager is to be accessed.

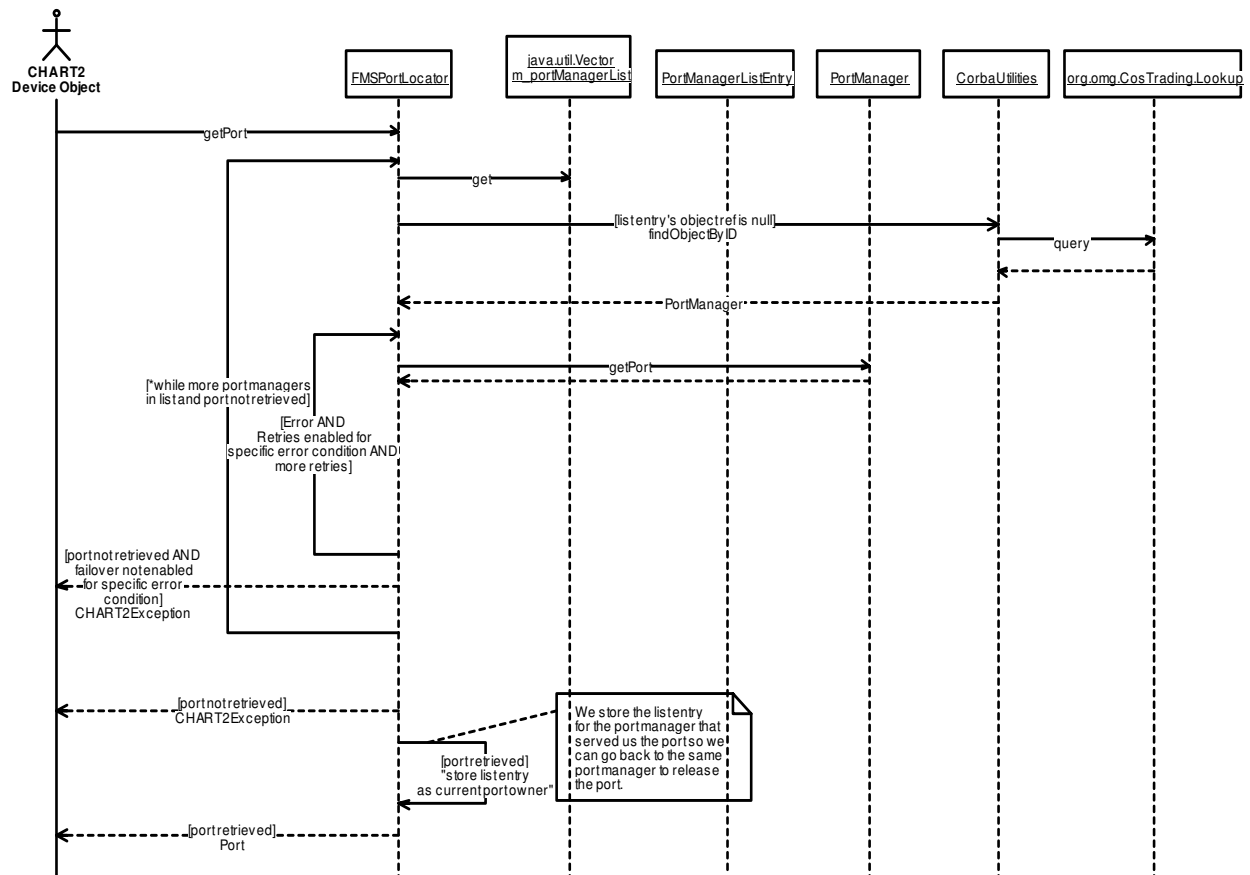


Figure 5-143 PortLocator:getPort (Sequence Diagram)

5.8 External Interface Module

5.8.1 Classes

5.8.1.1 DMSImportAcquireClasses (Class Diagram)

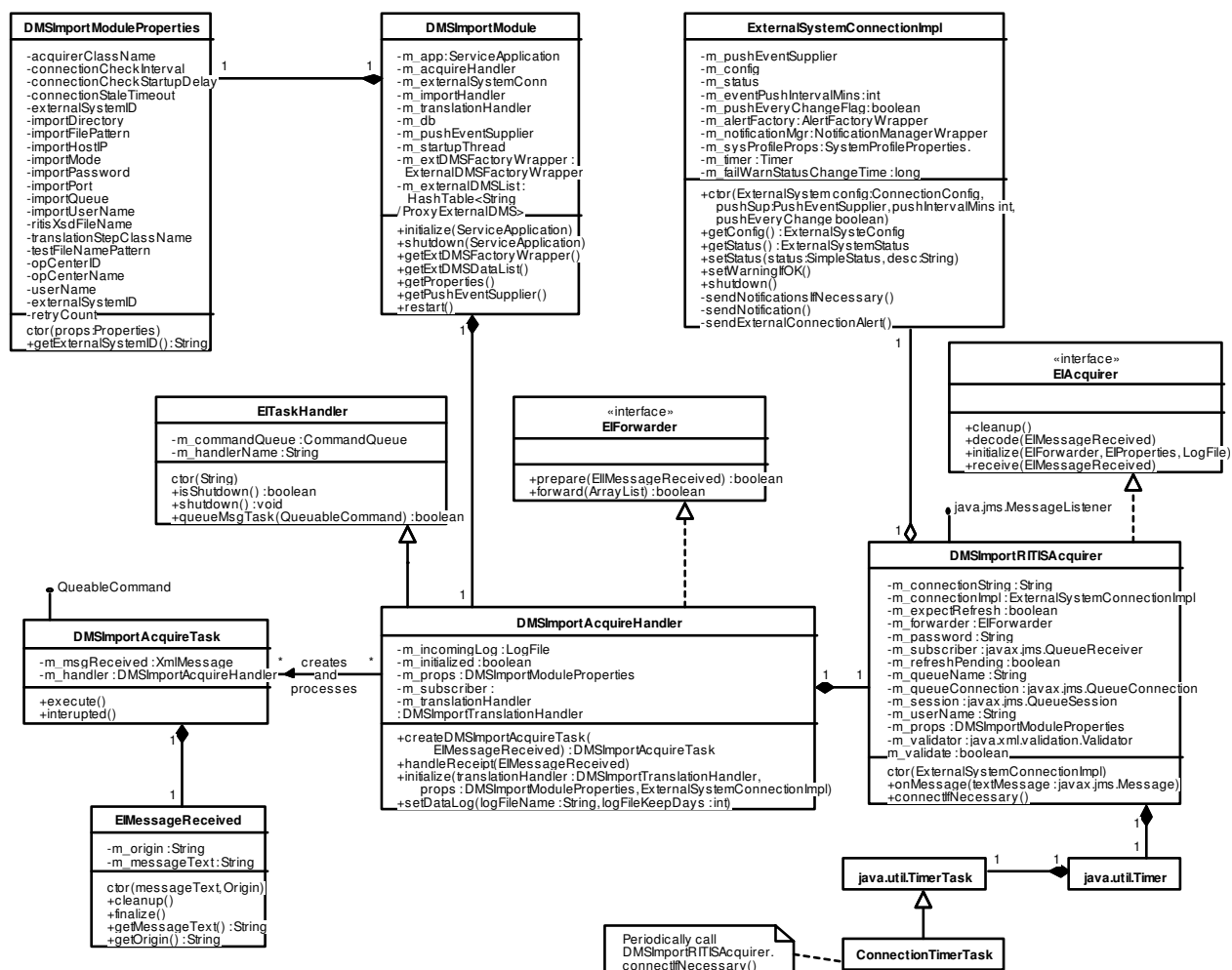


Figure 5-144 DMSImportAcquireClasses (Class Diagram)

5.8.1.1.1 ConnectionTimerTask (Class)

This class periodically checks the RITIS connection and reconnects if it's failed or stale (no activity for a period of time).

5.8.1.1.2 DMSImportAcquireHandler (Class)

This class handles data acquisition from the external source. It inherits CommandQueue processing functionality from the EITaskHandler. The custom Acquirer class, which

implements the EIAcquirer interface and is specified in the props, receives messages from the external system. When a message is received from the external system a DMSImportAcquireTask task is created and placed on a CommandQueue. When the CommandQueue executes the task it calls DMSImportAcquireHandler.handleReceipt() to process the message. The message is processed using the EIAcquirer interface and is forwarded for translation to the DMSImportTranslationHandler.

5.8.1.1.3 DMSImportAcquireTask (Class)

This class wraps an external DMS message so it can be put on a command queue. When the command queue calls the execute() method, this class invokes the DMSImportAcquireHandler to handle the message and prepare it for further processing (translation).

5.8.1.1.4 DMSImportModule (Class)

This module imports DMS data from an external source. The module allows for customizable (data source specific) data acquisition and translation using custom classes implementing generic interfaces). The module provides an ExternalDeviceManager interface used for configuring external DMS objects in chart and allows for ongoing updates to those objects.

5.8.1.1.5 DMSImportModuleProperties (Class)

This class holds all properties needed by the DMSImportModule including the name of the customized class used to acquire data from the external data source. Also, the name of the customized classes used to define the steps needed to translate the external DMS data into CHART DMS data.

5.8.1.1.6 DMSImportRITISAcquirer (Class)

This class knows how to connect to RITIS and obtain data on external DMS devices. It breaks the composite RITIS message into separate DMS inventory and DMS status messages and puts them on the translation command queue for processing. It obtains connection information from the DMSImportModuleProperties class.

5.8.1.1.7 EIAcquirer (Class)

Any class wishing to import data into CHART must support this EIAcquirer interface. The initialize method is called when the class wants to set up an external connection. The receive method is called when an external message is received for processing. The receive method should immediately place the external message on the acquirer command queue and be ready to take in the next external message. The decode method is called by the command queue to transform the external format into an internal format suitable for translation. The cleanup method is called before shutdown to give the implementing class an opportunity for a clean disconnect from the external source.

5.8.1.1.8 EIForwarder (Class)

This interface is implemented by classes that wish to process and forward messages for translation.

5.8.1.1.9 EIMessageReceived (Class)

This class holds the incoming message from the external source. Its associated task is quickly put on a command queue so the listener can get back to listening for new messages.

5.8.1.1.10 EITaskHandler (Class)

This class is the base class for the EI Task Handlers. It manages a command queue that is used to process EI tasks (CommandQueable) objects.

5.8.1.1.11 ExternalSystemConnectionImpl (Class)

This class knows how to maintain the status of external connections and push them up to the GUI. Also, ExternalSystemConnectionAlerts and Notifications can be sent as configured by the admin.

5.8.1.1.12 java.util.Timer (Class)

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

5.8.1.1.13 java.util.TimerTask (Class)

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

5.8.1.1.14 QueableCommand (Interface)

This interface is implemented by objects that can be placed on a command queue.

5.8.1.2 DMSImportChartClasses (Class Diagram)

This diagram shows the classes used to import external DMSs into CHART.

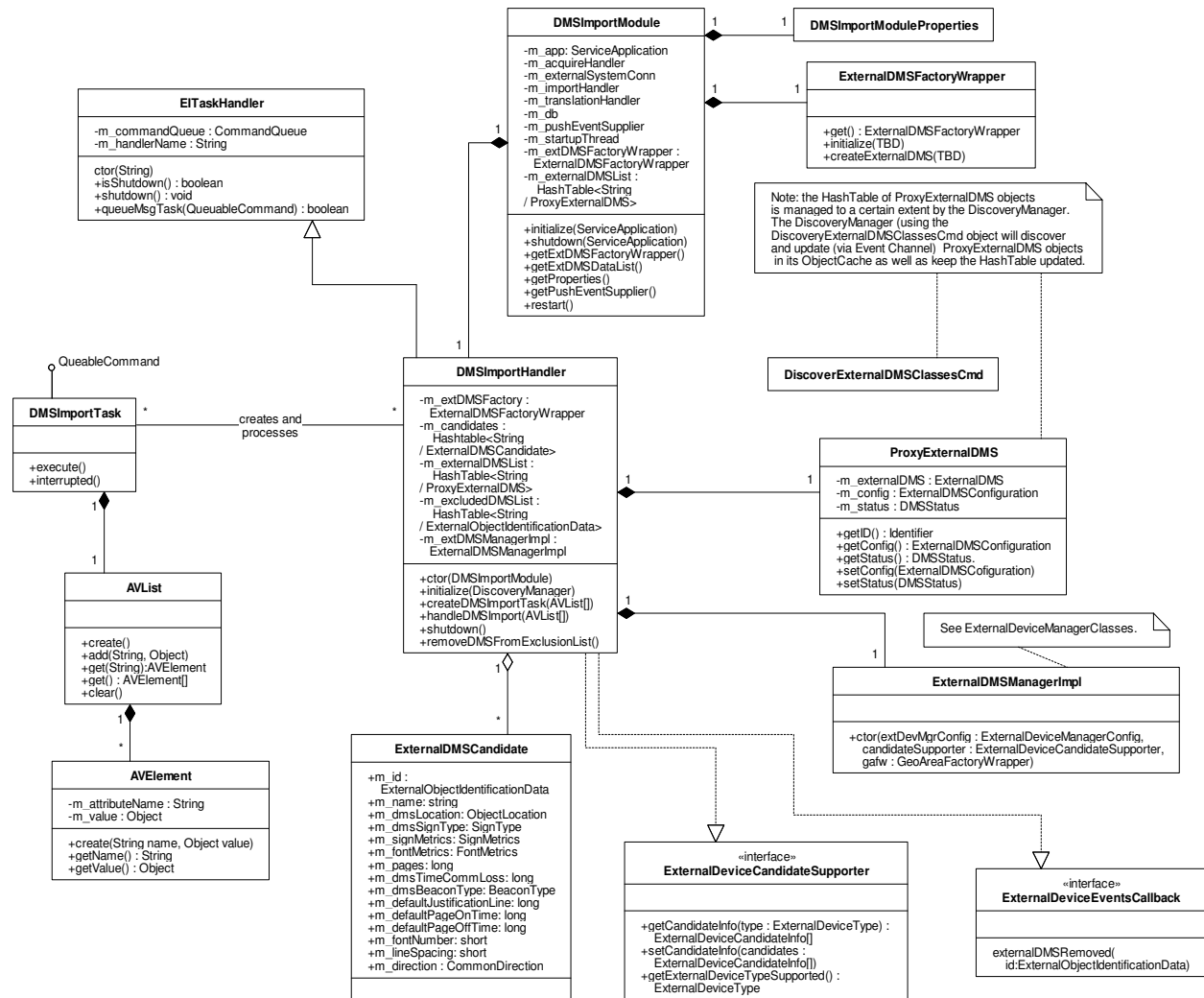


Figure 5-145 DMSImportChartClasses (Class Diagram)

5.8.1.2.1 AVElement (Class)

One element of an AVLList containing an attribute and a big-O-Object for storing/retrieving objects easily

5.8.1.2.2 AVLList (Class)

Generic collection of attribute and big-O-Object pairs used to move around the application.

5.8.1.2.3 DiscoverExternalDMSClassesCmd (Class)

This class is used by the DiscoveryManager to maintain the cache of ProxyExternalDMS objects used by the DMSImportModule.

5.8.1.2.4 DMSImportHandler (Class)

This class handles the actual import of the data into the CHART system proper. It inherits CommandQueue processing functionality from the EITaskHandler. This class maintains a cache of candidate DMSs from the external source and keeps a cache of references to ExternalDMS objects in CHART. The two caches are maintained via messages received from the external system and translated for CHART.

5.8.1.2.5 DMSImportModule (Class)

This module imports DMS data from an external source. The module allows for customizable (data source specific) data acquisition and translation using custom classes implementing generic interfaces). The module provides an ExternalDeviceManager interface used for configuring external DMS objects in chart and allows for ongoing updates to those objects.

5.8.1.2.6 DMSImportModuleProperties (Class)

This class holds all properties needed by the DMSImportModule including the name of the customized class used to acquire data from the external data source. Also, the name of the customized classes used to define the steps needed to translate the external DMS data into CHART DMS data.

5.8.1.2.7 DMSImportTask (Class)

This class is the command to process the external DMS import request received from the translation process and bound for CHART. It is executed by the CommandQueue asynchronously.

5.8.1.2.8 EITaskHandler (Class)

This class is the base class for the EI Task Handlers. It manages a command queue that is used to process EI tasks (CommandQueueable) objects.

5.8.1.2.9 ExternalDeviceCandidateSupporter (Class)

This interface is implemented by classes that are capable of supplying external device candidate data for the purposes of configuring external devices within CHART. Classes implementing this interface will also be capable of accepting administrator directives as to the disposition of external device candidates (i.e. specifically excluding or including candidate devices as "external" devices in CHART.

5.8.1.2.10 ExternalDeviceEventsCallback (Class)

This interface is used in conjunction with the DiscoveryExternalDMSClassCmd class which calls the appropriate interface method when certain corba events are received by the DiscoveryManager. Currently, only one method is supported. The externalDMSRemoved() method is call when a DMSRemoved corba event is received for an ExternalDMS object.

5.8.1.2.11 ExternalDMSCandidate (Class)

This class represents candidate DMS data from an external data source. It is populated with external data that has been translated into a CHART-centric structure.

5.8.1.2.12 ExternalDMSFactoryWrapper (Class)

This class finds a DMS Service and provides a facade for the creation of external DMS devices within CHART.

5.8.1.2.13 ExternalDMSManagerImpl (Class)

This class implements the ExternalDMSManager corba interface (which extends the ExternalDeviceManger corba interface). It extends the ExternalDeviceManagerImpl class and inherits most of its functionality from that base class.

5.8.1.2.14 ProxyExternalDMS (Class)

This class is used to cache external DMS device information. It holds and provides access to basic configuration and status data specific to an ExternalDMS type.

5.8.1.2.15 QueableCommand (Interface)

This interface is implemented by objects that can be placed on a command queue.

.

5.8.1.3 DMSImportModuleClasses (Class Diagram)

This diagram shows the implementation classes used to import external DMSs into CHART.

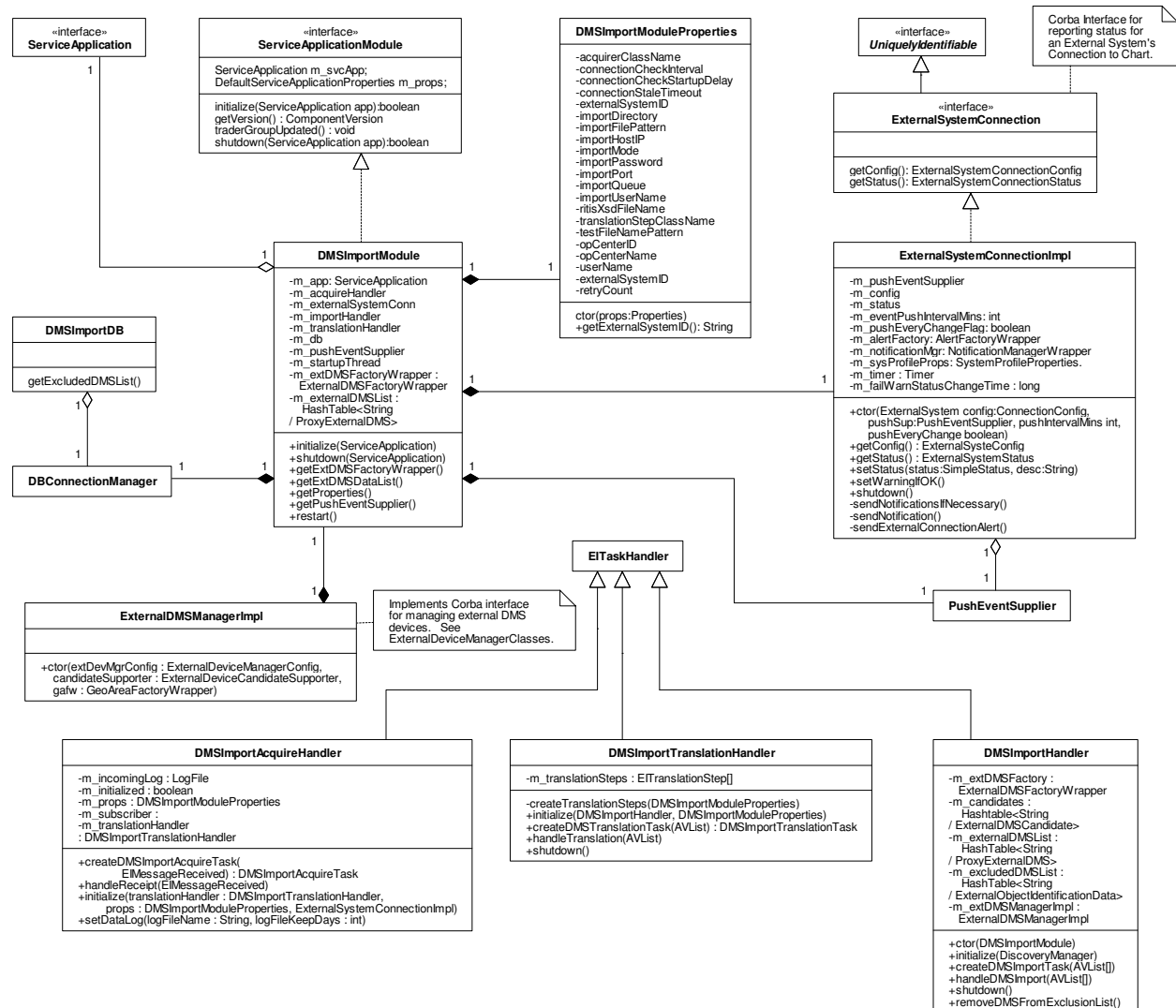


Figure 5-146 DMSImportModuleClasses (Class Diagram)

5.8.1.3.1 DBConnectionManager (Class)

This class implements a database connection manager that manages a pool of database connections. Any CHART II system thread requiring database access gets a database connection from the pool of connections maintained by this manager class. The connections are maintained in two separate lists namely, *inUseList* and *freeList*. The *inUseList* contains connections that have already been assigned to a thread. The *freeList* contains unassigned connections. This class assumes that an appropriate JDBC driver has been loaded either by using the "jdbc.drivers" system property or by loading it explicitly. The class has a monitor

thread that is started by the constructor. This connection monitor thread periodically checks the inuseList to see if there are connections that are owned by dead threads and move such connections to the freeList. The connection monitor thread is started only if a non-zero value is specified for the monitoring time interval in the constructor.

5.8.1.3.2 DMSImportAcquireHandler (Class)

This class handles data acquisition from the external source. It inherits CommandQueue processing functionality from the EITaskHandler. The custom Acquirer class, which implements the EIAcquirer interface and is specified in the props, receives messages from the external system. When a message is received from the external system a DMSImportAcquireTask task is created and placed on a CommandQueue. When the CommandQueue executes the task it calls DMSImportAcquireHandler.handleReceipt() to process the message. The message is processed using the EIAcquirer interface and is forwarded for translation to the DMSImportTranslationHandler.

5.8.1.3.3 DMSImportDB (Class)

This class provides a database interface for the DMSImportModule. It includes methods needed to store and retrieve external DMS related information. In particular, a list of excluded external DMS devices is kept.

5.8.1.3.4 DMSImportHandler (Class)

This class handles the actual import of the data into the CHART system proper. It inherits CommandQueue processing functionality from the EITaskHandler. This class maintains a cache of candidate DMSs from the external source and keeps a cache of references to ExternalDMS objects in CHART. The two caches are maintained via messages received from the external system and translated for CHART.

5.8.1.3.5 DMSImportModule (Class)

This module imports DMS data from an external source. The module allows for customizable (data source specific) data acquisition and translation using custom classes implementing generic interfaces). The module provides an ExternalDeviceManager interface used for configuring external DMS objects in chart and allows for ongoing updates to those objects.

5.8.1.3.6 DMSImportModuleProperties (Class)

This class holds all properties needed by the DMSImportModule including the name of the customized class used to acquire data from the external data source. Also, the name of the customized classes used to define the steps needed to translate the external DMS data into CHART DMS data.

5.8.1.3.7 DMSImportTranslationHandler (Class)

This class creates and handles DMSImportTranslationTasks. It inherits CommandQueue

processing functionality from the EITaskHandler. It handles translation step classes that are specified in the props file and customized for the specific translation needed for the data provided by the external system. After translation the data is forwarded to the DMSImportHandler for import into CHART.

5.8.1.3.8 EITaskHandler (Class)

This class is the base class for the EI Task Handlers. It manages a command queue that is used to process EI tasks (CommandQueueable) objects.

5.8.1.3.9 ExternalDMSManagerImpl (Class)

This class implements the ExternalDMSManager corba interface (which extends the ExternalDeviceManger corba interface). It extends the ExternalDeviceManagerImpl class and inherits most of its functionality from that base class.

5.8.1.3.10 ExternalSystemConnection (Class)

This interface defines external connections and provides status reporting.

5.8.1.3.11 ExternalSystemConnectionImpl (Class)

This class knows how to maintain the status of external connections and push them up to the GUI. Also, ExternalSystemConnectionAlerts and Notifications can be sent as configured by the admin.

5.8.1.3.12 PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.8.1.3.13 ServiceApplication (Class)

This interface is implemented by objects that can provide the basic services needed by a ChartII service application. These services include providing access to basic CORBA objects that are needed by service applications, such as the ORB, POA, Trader, and Event Service.

5.8.1.3.14 ServiceApplicationModule (Class)

This interface is implemented by modules that serve CORBA objects. Implementing

classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

5.8.1.3.15 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

.

5.8.1.4 DMSImportTranslationClasses (Class Diagram)

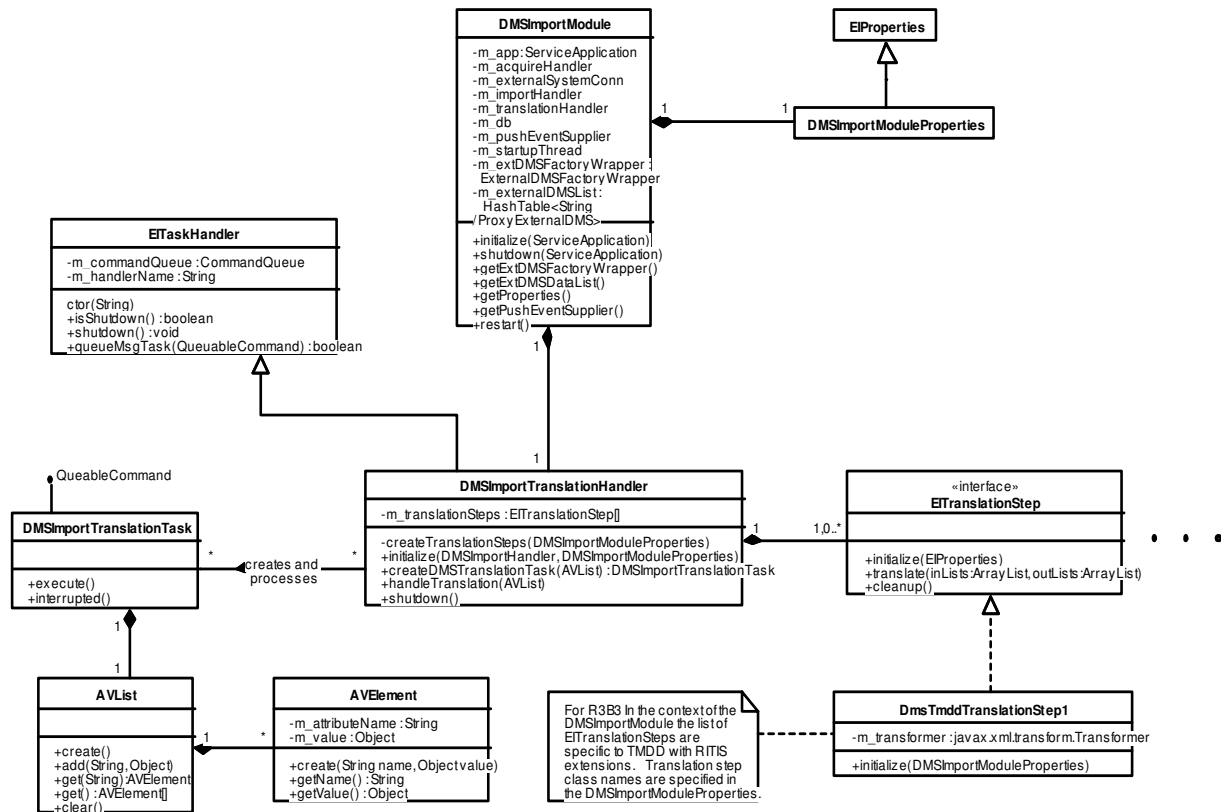


Figure 5-147 DMSImportTranslationClasses (Class Diagram)

5.8.1.4.1 AVElement (Class)

One element of an AVLList containing an attribute and a big-O-Object for storing/retrieving objects easily

5.8.1.4.2 AVLList (Class)

Generic collection of attribute and big-O-Object pairs used to move around the application.

5.8.1.4.3 DMSImportModule (Class)

This module imports DMS data from an external source. The module allows for customizable (data source specific) data acquisition and translation using custom classes implementing generic interfaces). The module provides an ExternalDeviceManager interface used for configuring external DMS objects in chart and allows for ongoing updates to those objects.

5.8.1.4.4 DMSImportModuleProperties (Class)

This class holds all properties needed by the DMSImportModule including the name of the customized class used to acquire data from the external data source. Also, the name of the customized classes used to define the steps needed to translate the external DMS data into CHART DMS data.

5.8.1.4.5 DMSImportTranslationHandler (Class)

This class creates and handles DMSImportTranslationTasks. It inherits CommandQueue processing functionality from the EITaskHandler. It handles translation step classes that are specified in the props file and customized for the specific translation needed for the data provided by the external system. After translation the data is forwarded to the DMSImportHandler for import into CHART.

5.8.1.4.6 DMSImportTranslationTask (Class)

This class wraps an external DMS inventory or status message so it can be put on a command queue. When the command queue calls the execute() method, this class invokes the DMSImportTranslationHandler to translate it into CHART-centric DMS data. .

5.8.1.4.7 DmsTmddTranslationStep1 (Class)

This class represents step one of a one step translation which will translate TMDD DMS inventory and DMS status messages to CHART-centric terms. An example of when a multiple step translation may be needed is when a complex message needs to be simplified before further translation can be done. Multiple simple steps may be able to accomplish what one complex step could. For ease of maintenance, multiple steps may be better in that case.

5.8.1.4.8 EIProperties (Class)

This class supports properties that are generic to all External Interface modules such as log filenames.

5.8.1.4.9 EITaskHandler (Class)

This class is the base class for the EI Task Handlers. It manages a command queue that is used to process EI tasks (CommandQueueable) objects.

5.8.1.4.10 EITranslationStep (Class)

An EITranslationStep is the base class for all translations. Implementing translations define how to translate from a particular type of AVLList to another particular type of AVLList.

5.8.1.4.11 QueableCommand (Interface)

This interface is implemented by objects that can be placed on a command queue.

5.8.1.5 EventAtisImportChartClasses (Class Diagram)

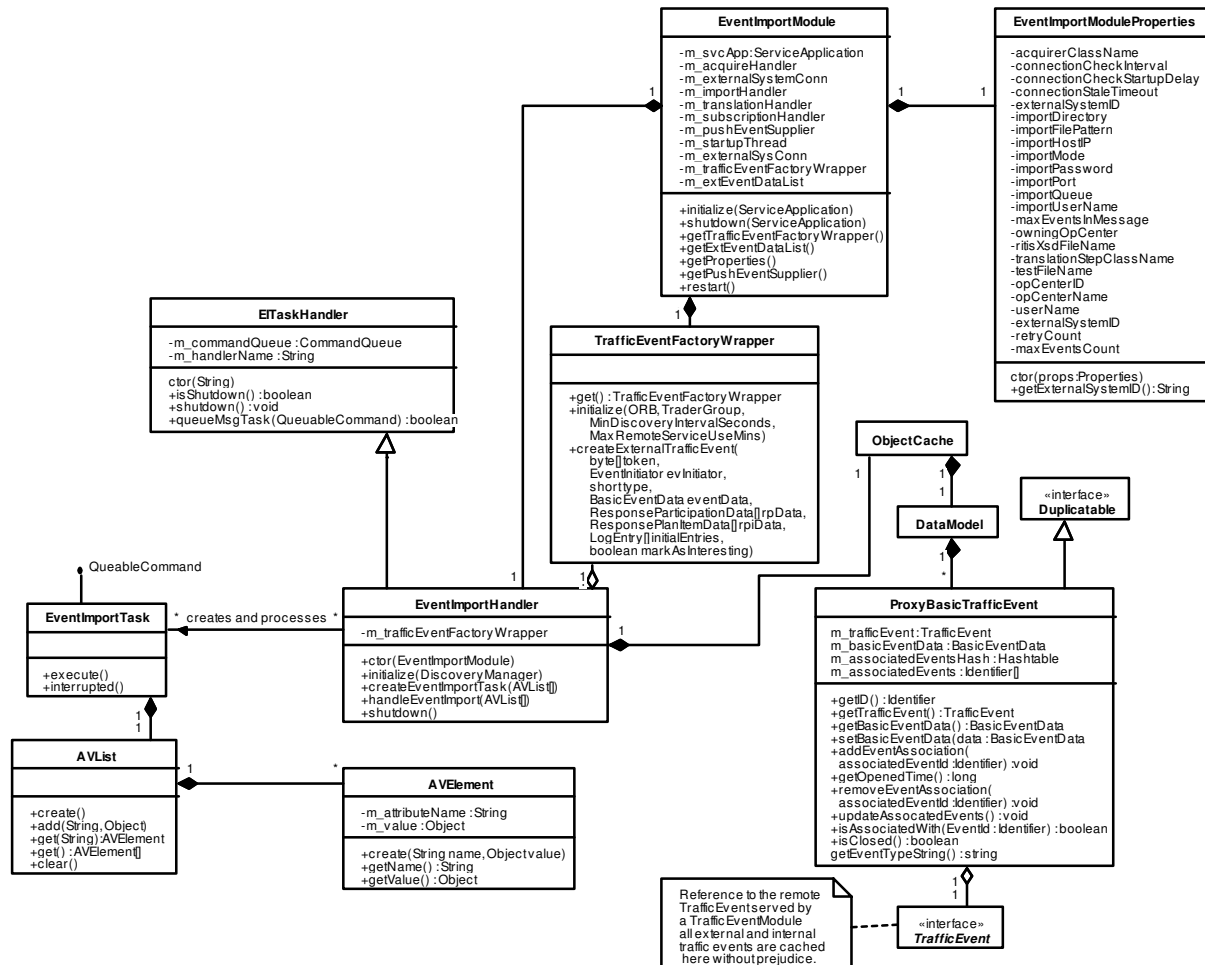


Figure 5-148 EventAtisImportChartClasses (Class Diagram)

5.8.1.5.1 AVElement (Class)

One element of an AVLList containing an attribute and a big-O-Object for storing/retrieving objects easily

5.8.1.5.2 AVLList (Class)

Generic collection of attribute and big-O-Object pairs used to move around the application.

5.8.1.5.3 DataModel (Class)

The data model class serves as a collection of objects. It provides an efficient lookup mechanism for locating any object, and methods which allow for the retrieval of all objects of a particular type. Additionally, this class provides the ability to attach observer objects

which are notified when objects are added to or removed from the model. Objects may also notify the DataModel that they have been modified. The model will periodically notify all attached observers of the changes to objects in the model.

5.8.1.5.4 Duplicatable (Class)

This java interface is implemented by classes which have sense of being "duplicated" within the CHART system. This allows the ObjectCache to search for duplicates of any Duplicatable object. This is different from "equals()" or "compareTo()". To cite two examples: Alerts within CHART are duplicates if they refer to the same objects within CHART (but do not have the same Alert ID, which is more closely associated with "equals()"). Traffic Events within CHART are duplicates if they have the same location (but do not have the same Traffic Event ID).

5.8.1.5.5 EITaskHandler (Class)

This class is the base class for the EI Task Handlers. It manages a command queue that is used to process EI tasks (CommandQueueable) objects.

5.8.1.5.6 EventImportHandler (Class)

This class is responsible for handling EI tasks representing external CHART events. It is called when an EventImportTask is put on its queue. It then converts the AVLlist to the CHART event components needed to create a CHART event.

5.8.1.5.7 EventImportModule (Class)

This module imports traffic events from an external source by loading the class specified in its props file. The class knows how to connect to an external traffic event source and how to prepare it for translation. A separate property tells the module how to translate the external traffic event into an AVLlist that directly maps to a CHART traffic event.

5.8.1.5.8 EventImportModuleProperties (Class)

This class holds all properties needed by the ExternalImportModule including the name of the class to be loaded to make the external connection and the steps needed to translate the external traffic event into an internal traffic event.

5.8.1.5.9 EventImportTask (Class)

This class is the command to process the event import request received from the translation process and bound for CHART. It is executed by the CommandQueue asynchronously.

5.8.1.5.10 ObjectCache (Class)

The ObjectCache is a wrapper for the DataModel. It provides access to DataModel methods to find objects in the data model, delegating those methods to the DataModel itself. It also provides additional methods of finding name filtered objects and discovering "duplicate" objects (as defined by an isDuplicateOf() method of the Duplicatable interface).

5.8.1.5.11 ProxyBasicTrafficEvent (Class)

This class is used as a proxy for traffic events existing in all traffic event services (including the local service). The proxy traffic events cached are not complete copies of the traffic events, because the full range of data is not needed. The ProxyBasicTrafficEvent data consists of BasicEventData and associated events only (this is why the names of these objects contain the word "Basic", e.g., DiscoverBasicTrafficEventClassesCommand. These proxy traffic events allow every traffic event service in the system to have some knowledge of every traffic event in the entire system, for the purpose of detecting duplicate traffic events.

5.8.1.5.12 QueableCommand (Interface)

This interface is implemented by objects that can be placed on a command queue.

5.8.1.5.13 TrafficEvent (Class)

Objects of this type represent traffic events that require action from system operators.

5.8.1.5.14 TrafficEventFactoryWrapper (Class)

This class finds a Traffic Event Service and provides a facade to it for the creation of external traffic events.

5.8.1.6 ExternalDeviceManagerClasses (Class Diagram)

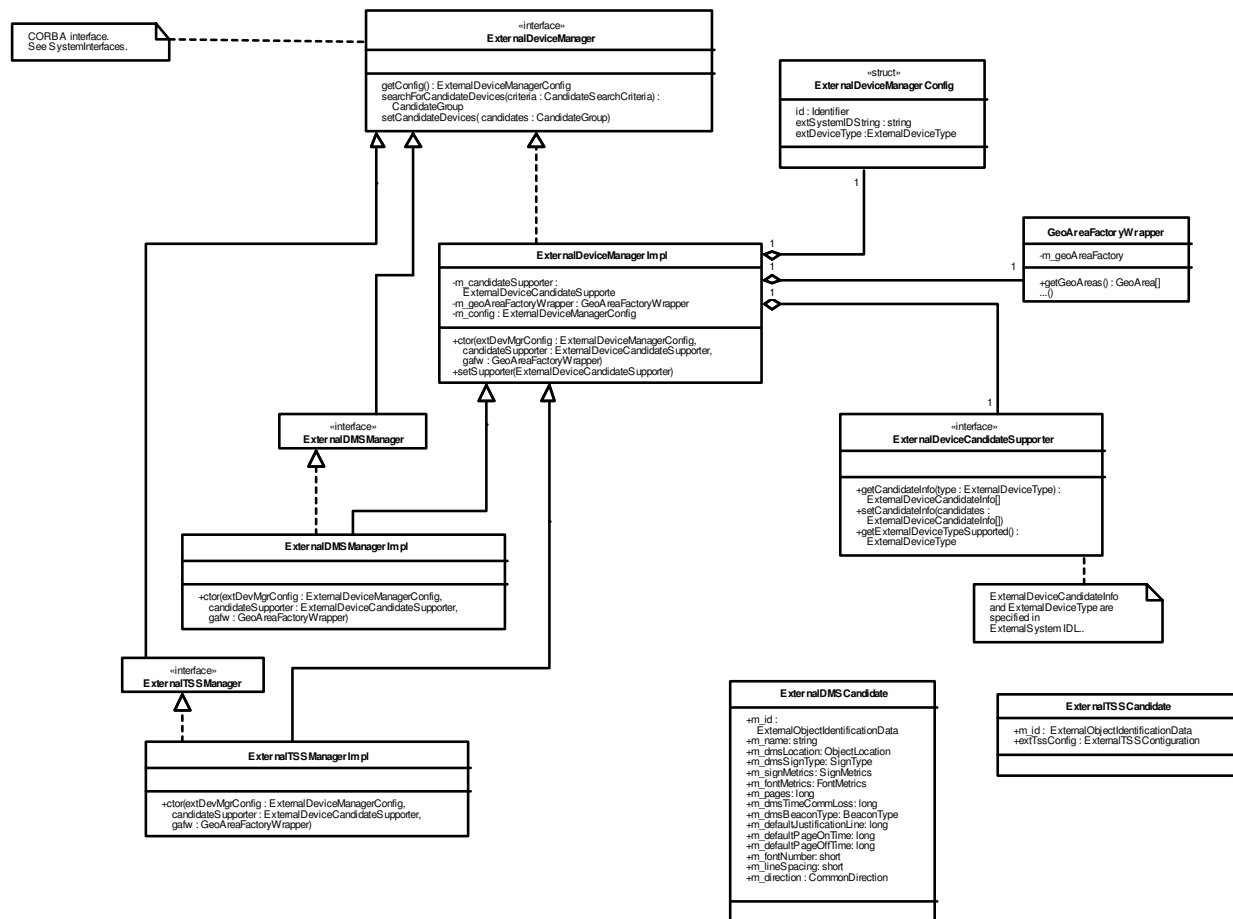


Figure 5-149 ExternalDeviceManagerClasses (Class Diagram)

5.8.1.6.1 ExternalDeviceCandidateSupporter (Class)

This interface is implemented by classes that are capable of supplying external device candidate data for the purposes of configuring external devices within CHART. Classes implementing this interface will also be capable of accepting administrator directives as to the disposition of external device candidates (i.e. specifically excluding or including candidate devices as "external" devices in CHART).

5.8.1.6.2 ExternalDeviceManager (Class)

This interface defines operations used for configuring External Devices in CHART. Each ExternalDeviceManager manages external devices of a specific type (DMS, TSS, ...) for a specific External System (i.e. data source).

5.8.1.6.3 ExternalDeviceManagerConfig (Class)

This struct defines the configuration for an ExternalDeviceManager in CHART.

5.8.1.6.4 ExternalDeviceManagerImpl (Class)

This class implements the ExternalDeviceManager corba interface. It provides the functionality used by the GUI to allow administrators to configure external devices in CHART. It provides for the searching of external data source's candidate devices. It also provides the interface, thru the GUI, for administrators to select candidate devices for inclusion in CHART (I.E. import the device into CHART).

5.8.1.6.5 ExternalDMSCandidate (Class)

This class represents candidate DMS data from an external data source. It is populated with external data that has been translated into a CHART-centric structure.

5.8.1.6.6 ExternalDMSManager (Class)

This is a naming interface (empty interface) used to specify a external device manager specific to DMS devices. Its used as a convenience when querying traders and to clarify implementation.

5.8.1.6.7 ExternalDMSManagerImpl (Class)

This class implements the ExternalDMSManager corba interface (which extends the ExternalDeviceManger corba interface). It extends the ExternalDeviceManagerImpl class and inherits most of its functionality from that base class.

5.8.1.6.8 ExternalTSSCandidate (Class)

This class represents candidate TSS data from an external data source. It is populated with external data that has been translated into a CHART-centric structure.

5.8.1.6.9 ExternalTSSManager (Class)

This is a naming interface (empty interface) used to specify a external device manager specific to TSS devices. Its used as a convenience when querying traders and to clarify implementation.

5.8.1.6.10 ExternalTSSManagerImpl (Class)

This class implements the ExternalTSSManager corba interface (which extends the ExternalDeviceManger corba interface). It extends the ExternalDeviceManagerImpl class and inherits most of its functionality from that base class.

5.8.1.6.11 GeoAreaFactoryWrapper (Class)

This singleton class presents the same interface as the GeoAreaFactory, but uses a FirstAvailableOfferWrapper to provide fault tolerant access to the methods.

5.8.1.7 ExternalSystemConnectionClasses (Class Diagram)

The ExternalSystemConnection class diagram list classes involved with External System Connection monitoring and status reporting. The ExternalSystemConnectionImpl implements the ExternalSystemConnection interface allowing clients to retrieve configuration and status info for the connection being represented. An AlertFactoryWrapper and NotificationManagerWrapper are used to send ExternalSystemConnection Alerts as configured to do so. Access to SystemProfileProperties is provided by the SystemProfileProperties class.

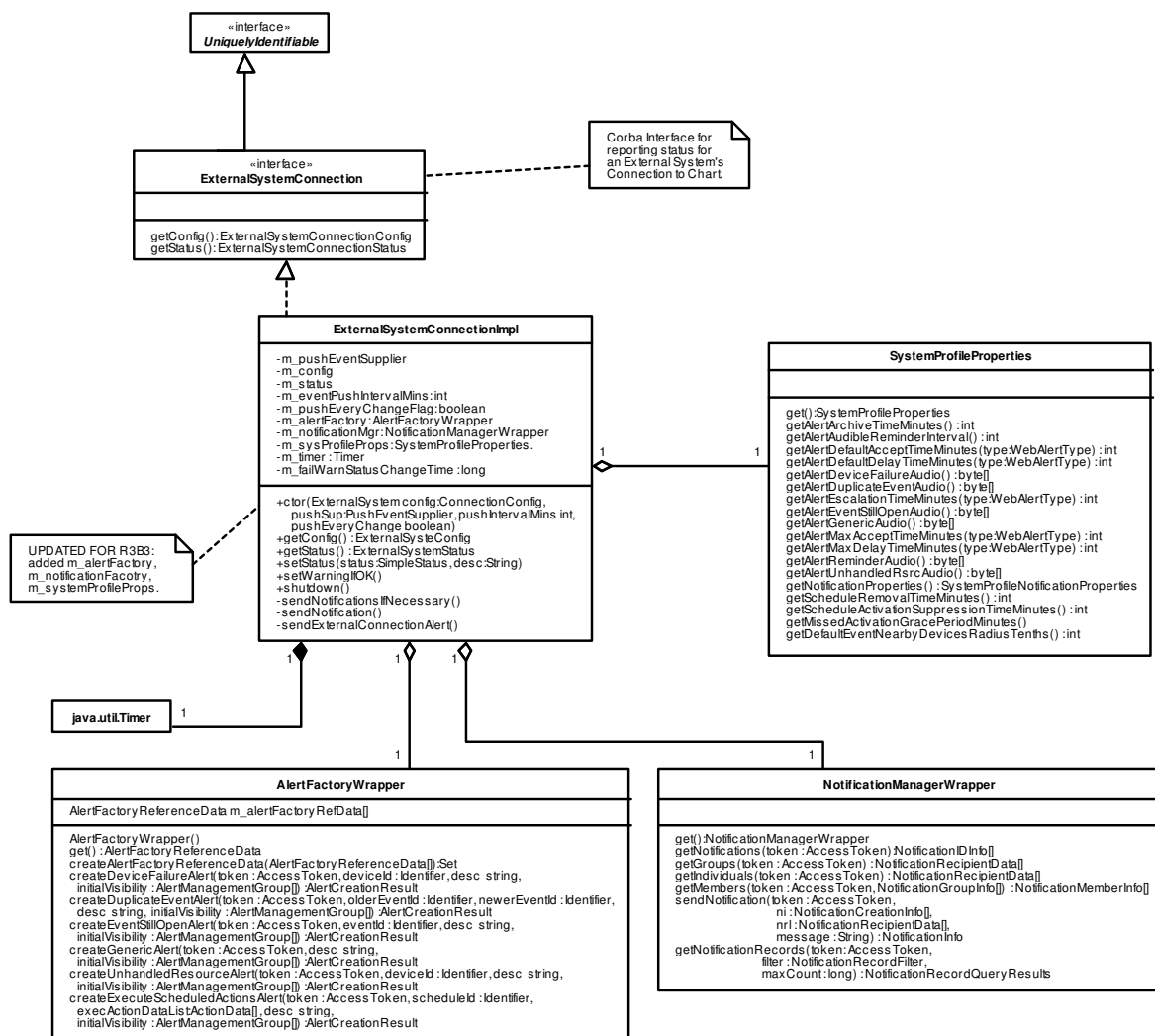


Figure 5-150 ExternalSystemConnectionClasses (Class Diagram)

5.8.1.7.1 AlertFactoryWrapper (Class)

This singleton class provides a wrapper for the Alert Factory that provides automatic

location of an Alert Factory and automatic re-discovery should the Alert Factory reference return an error. This class also allows for built-in fault tolerance by automatically failing over to a "working" Alert Factory without the user of this class being aware that this is being done. In addition, this class defers the discovery of the Alert Factory until its first use, thus eliminating a start-up dependency for modules that use the Alert Factory.

This class delegates all of its method calls to the system AlertFactory using its currently known good reference to an AlertFactory. If the current reference returns a CORBA failure in the delegated call, this class automatically switches to another reference. When there are no good references (as is true the first time the object is used), this class issues a trader query to (re)discover the published Alert Factory objects in the system. During a method call, the trader will be queried at most one time and under normal circumstances, not at all.

5.8.1.7.2 ExternalSystemConnection (Class)

This interface defines external connections and provides status reporting.

5.8.1.7.3 ExternalSystemConnectionImpl (Class)

This class knows how to maintain the status of external connections and push them up to the GUI. Also, ExternalSystemConnectionAlerts and Notifications can be sent as configured by the admin.

5.8.1.7.4 java.util.Timer (Class)

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

5.8.1.7.5 NotificationManagerWrapper (Class)

This singleton class presents the same interface as the NotificationManager, but uses a FirstAvailableOfferWrapper to provide fault tolerant access to the methods.

5.8.1.7.6 SystemProfileProperties (Class)

This class is used to cache the system profile properties and provide access to them. It is also used to interact with the server to change system profile settings.

5.8.1.7.7 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.8.1.8 TSSImportAcquireClasses (Class Diagram)

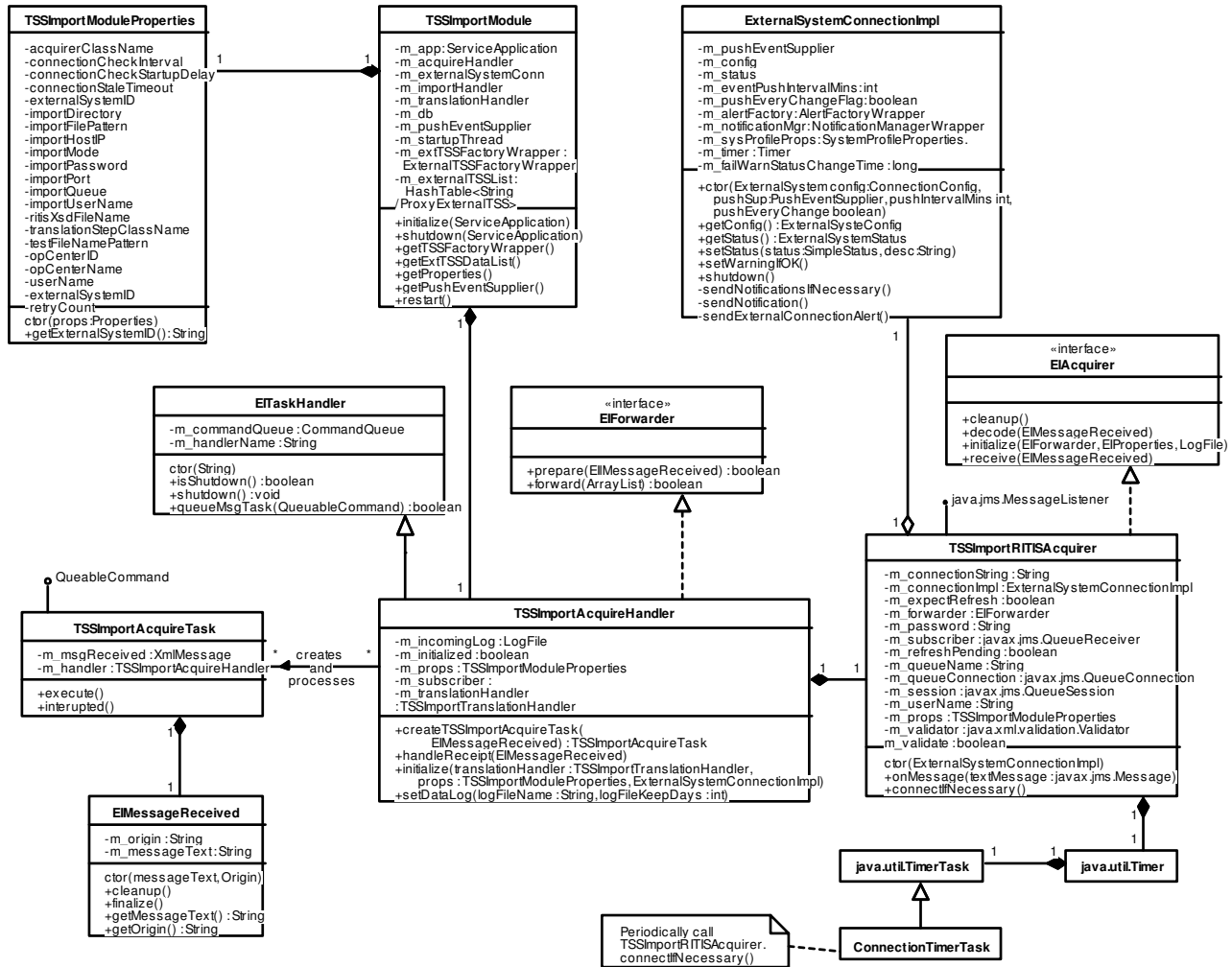


Figure 5-151 TSSImportAcquireClasses (Class Diagram)

5.8.1.8.1 ConnectionTimerTask (Class)

This class periodically checks the RITIS connection and reconnects if it's failed or stale (no activity for a period of time).

5.8.1.8.2 EIAcquirer (Class)

Any class wishing to import data into CHART must support this EIAcquirer interface. The initialize method is called when the class wants to set up an external connection. The receive method is called when an external message is received for processing. The receive method should immediately place the external message on the acquirer command queue and be ready to take in the next external message. The decode method is called by the command queue to transform the external format into an internal format suitable for

translation. The cleanup method is called before shutdown to give the implementing class an opportunity for a clean disconnect from the external source.

5.8.1.8.3 EIForwarder (Class)

This interface is implemented by classes that wish to process and forward messages for translation.

5.8.1.8.4 IMessageReceived (Class)

This class holds the incoming message from the external source. Its associated task is quickly put on a command queue so the listener can get back to listening for new messages.

5.8.1.8.5 EITaskHandler (Class)

This class is the base class for the EI Task Handlers. It manages a command queue that is used to process EI tasks (CommandQueueable) objects.

5.8.1.8.6 ExternalSystemConnectionImpl (Class)

This class knows how to maintain the status of external connections and push them up to the GUI. Also, ExternalSystemConnectionAlerts and Notifications can be sent as configured by the admin.

5.8.1.8.7 java.jms.MessageListener (Interface)

The MessageListener class is an interface that supports the receipt of messages from a JMS queue. Implementers must handle the onMessage() method for all messages and optionally support the onException() method when there is a problem with the JMS connection.

5.8.1.8.8 java.util.Timer (Class)

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

5.8.1.8.9 java.util.TimerTask (Class)

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

5.8.1.8.10 QueableCommand (Interface)

This interface is implemented by objects that can be placed on a command queue.

5.8.1.8.11 TSSImportAcquireHandler (Class)

This class handles data acquisition from the external source. It inherits CommandQueue processing functionality from the EITaskHandler. The custom Acquirer class, which implements the EIAcquirer interface and is specified in the props, receives messages from the external system. When a message is received from the external system a

TSSImportAcquireTask task is created and placed on a CommandQueue. When the CommandQueue executes the task it calls TSSImportAcquireHandler.handleReceipt() to process the message. The message is processed using the EIAcquirer interface and is forwarded for translation to the TSSImportTranslationHandler.

5.8.1.8.12 TSSImportAcquireTask (Class)

This class wraps an external TSS message so it can be put on a command queue. When the command queue calls the execute() method, this class invokes the TSSImportAcquireHandler to handle the message and prepare it for further processing (translation).

5.8.1.8.13 TSSImportModule (Class)

This module imports TSS data from an external source. The module allows for customizable (data source specific) data acquisition and translation using custom classes implementing generic interfaces). The module provides an ExternalDeviceManager interface used for configuring external TSS objects in chart and allows for ongoing updates to those objects.

5.8.1.8.14 TSSImportModuleProperties (Class)

This class holds all properties needed by the TSSImportModule including the name of the customized class used to acquire data from the external data source. Also, the name of the customized classes used to define the steps needed to translate the external TSS data into CHART TSS data.

5.8.1.8.15 TSSImportRITISAcquirer (Class)

This class knows how to connect to RITIS and obtain data on external TSS devices. It breaks the composite RITIS message into separate TSS inventory and TSS status messages and puts them on the translation command queue for processing. It obtains connection information from the TSSImportModuleProperties class.

5.8.1.9 TSSImportChartClasses (Class Diagram)

This diagram shows the classes used to import external TSSs into CHART.

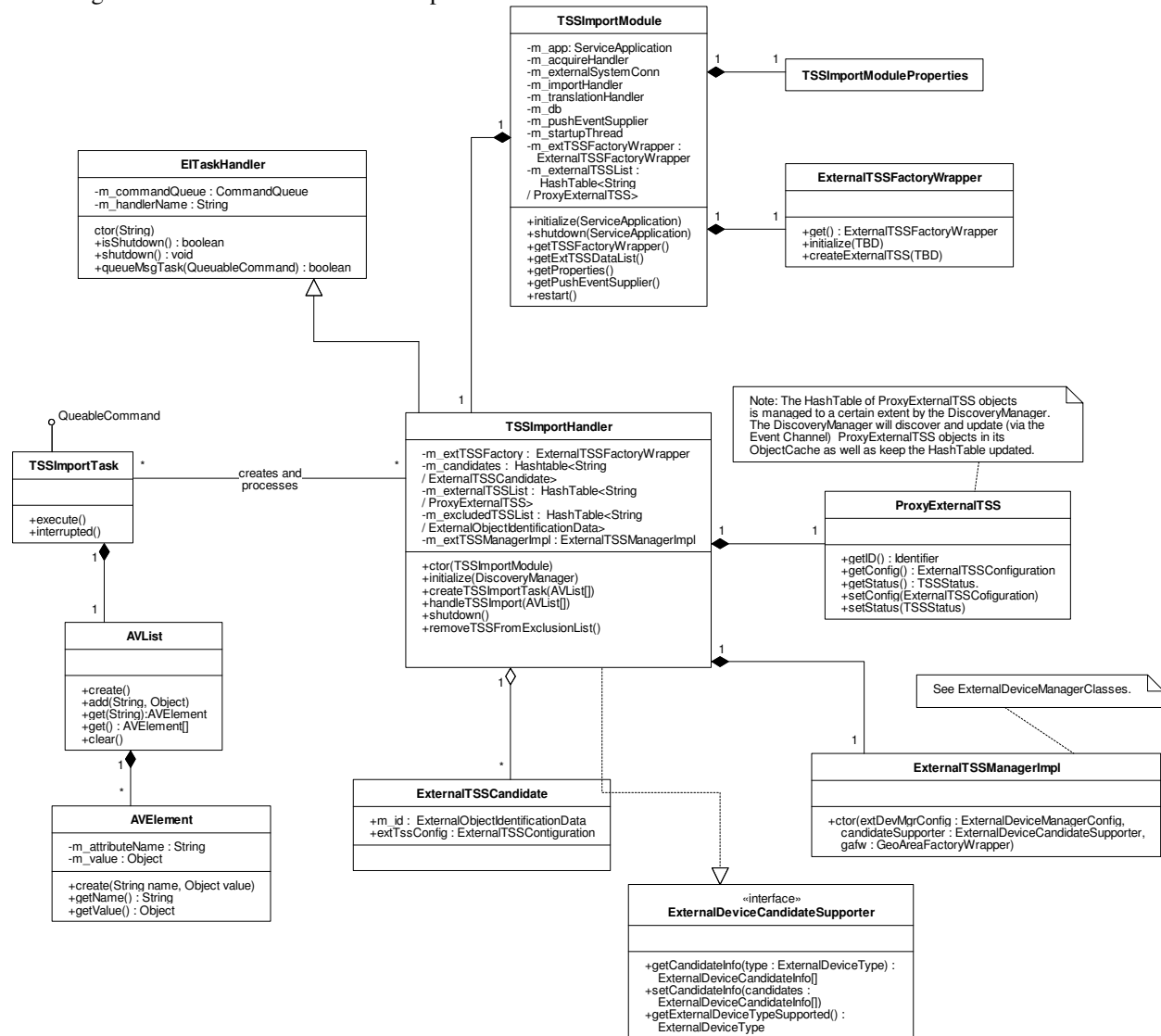


Figure 5-152 TSSImportChartClasses (Class Diagram)

5.8.1.9.1 AVElement (Class)

One element of an AVLList containing an attribute and a big-O-Object for storing/retrieving objects easily

5.8.1.9.2 AVLList (Class)

Generic collection of attribute and big-O-Object pairs used to move around the application.

5.8.1.9.3 EITaskHandler (Class)

This class is the base class for the EI Task Handlers. It manages a command queue that is used to process EI tasks (CommandQueable) objects.

5.8.1.9.4 ExternalDeviceCandidateSupporter (Class)

This interface is implemented by classes that are capable of supplying external device candidate data for the purposes of configuring external devices within CHART. Classes implementing this interface will also be capable of accepting administrator directives as to the disposition of external device candidates (i.e. specifically excluding or including candidate devices as "external" devices in CHART.

5.8.1.9.5 ExternalTSSCandidate (Class)

This class represents candidate TSS data from an external data source. It is populated with external data that has been translated into a CHART-centric structure.

5.8.1.9.6 ExternalTSSFactoryWrapper (Class)

This class finds a TSS Service and provides a facade for the creation of external TSS devices within CHART.

5.8.1.9.7 ExternalTSSManagerImpl (Class)

This class implements the ExternalTSSManager corba interface (which extends the ExternalDeviceManger corba interface). It extends the ExternalDeviceManagerImpl class and inherits most of its functionality from that base class.

5.8.1.9.8 ProxyExternalTSS (Class)

This class is used to cache external TSS device information. It holds and provides access to basic configuration and status data specific to an ExternalTSS type.

5.8.1.9.9 QueableCommand (Interface)

This interface is implemented by objects that can be placed on a command queue.

5.8.1.9.10 TSSImportHandler (Class)

This class handles the actual import of the data into the CHART system proper. It inherits CommandQueue processing functionality from the EITaskHandler. This class maintains a cache of candidate TSSs from the external source and keeps a cache of references to ExternalTSS objects in CHART. The two caches are maintained via messages received from the external system and translated for CHART.

5.8.1.9.11 TSSImportModule (Class)

This module imports TSS data from an external source. The module allows for customizable (data source specific) data acquisition and translation using custom classes

implementing generic interfaces). The module provides an ExternalDeviceManager interface used for configuring external TSS objects in chart and allows for ongoing updates to those objects.

5.8.1.9.12 TSSImportModuleProperties (Class)

This class holds all properties needed by the TSSImportModule including the name of the customized class used to acquire data from the external data source. Also, the name of the customized classes used to define the steps needed to translate the external TSS data into CHART TSS data.

5.8.1.9.13 TSSImportTask (Class)

This class is the command to process the external TSS import request received from the translation process and bound for CHART. It is executed by the CommandQueue asynchronously.

5.8.1.10 TSSImportModuleClasses (Class Diagram)

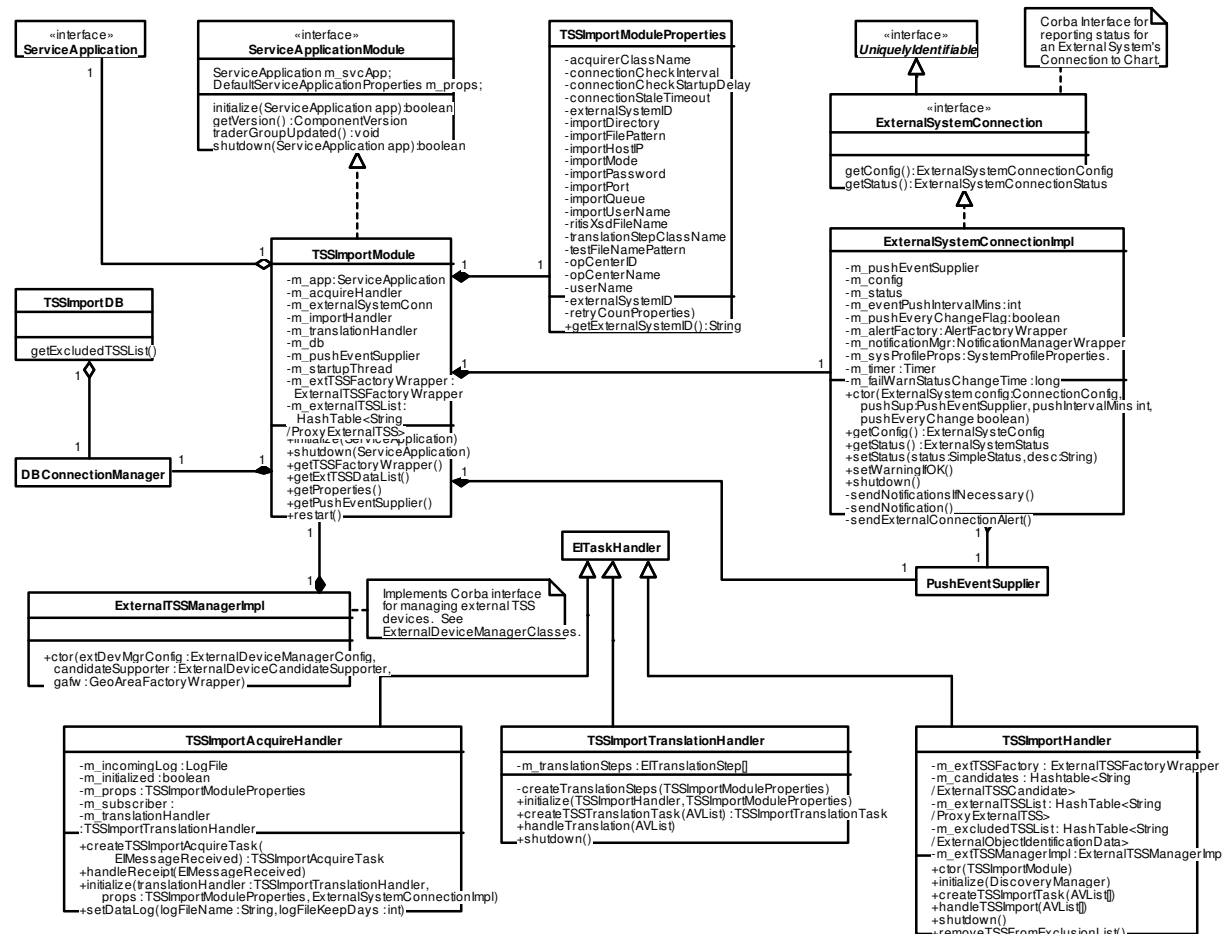


Figure 5-153 TSSImportModuleClasses (Class Diagram)

5.8.1.10.1 DBConnectionManager (Class)

This class implements a database connection manager that manages a pool of database connections. Any CHART II system thread requiring database access gets a database connection from the pool of connections maintained by this manager class. The connections are maintained in two separate lists namely, inUseList and freeList. The inUseList contains connections that have already been assigned to a thread. The freeList contains unassigned connections. This class assumes that an appropriate JDBC driver has been loaded either by using the "jdbc.drivers" system property or by loading it explicitly. The class has a monitor thread that is started by the constructor. This connection monitor thread periodically checks the inuseList to see if there are connections that are owned by dead threads and move such connections to the freeList. The connection monitor thread is started only if a non-zero value is specified for the monitoring time interval in the constructor.

5.8.1.10.2 EITaskHandler (Class)

This class is the base class for the EI Task Handlers. It manages a command queue that is used to process EI tasks (CommandQueueable) objects.

5.8.1.10.3 ExternalSystemConnection (Class)

This interface defines external connections and provides status reporting.

5.8.1.10.4 ExternalSystemConnectionImpl (Class)

This class knows how to maintain the status of external connections and push them up to the GUI. Also, ExternalSystemConnectionAlerts and Notifications can be sent as configured by the admin.

5.8.1.10.5 ExternalTSSManagerImpl (Class)

This class implements the ExternalTSSManager corba interface (which extends the ExternalDeviceManger corba interface). It extends the ExternalDeviceManagerImpl class and inherits most of its functionality from that base class.

5.8.1.10.6 PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.8.1.10.7 ServiceApplication (Class)

This interface is implemented by objects that can provide the basic services needed by a ChartII service application. These services include providing access to basic CORBA objects that are needed by service applications, such as the ORB, POA, Trader, and Event Service.

5.8.1.10.8 ServiceApplicationModule (Class)

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

5.8.1.10.9 TSSImportAcquireHandler (Class)

This class handles data acquisition from the external source. It inherits CommandQueue processing functionality from the EITaskHandler. The custom Acquirer class, which implements the EIAcquirer interface and is specified in the props, receives messages from the external system. When a message is received from the external system a TSSImportAcquireTask task is created and placed on a CommandQueue. When the CommandQueue executes the task it calls TSSImportAcquireHandler.handleReceipt() to process the message. The message is processed using the EIAcquirer interface and is forwarded for translation to the TSSImportTranslationHandler.

5.8.1.10.10 TSSImportDB (Class)

This class provides a database interface for the TSSImportModule. It includes methods needed to store and retrieve external TSS related information. In particular, a list of excluded external TSS devices is kept.

5.8.1.10.11 TSSImportHandler (Class)

This class handles the actual import of the data into the CHART system proper. It inherits CommandQueue processing functionality from the EITaskHandler. This class maintains a cache of candidate TSSs from the external source and keeps a cache of references to ExternalTSS objects in CHART. The two caches are maintained via messages received from the external system and translated for CHART.

5.8.1.10.12 TSSImportModule (Class)

This module imports TSS data from an external source. The module allows for customizable (data source specific) data acquisition and translation using custom classes implementing generic interfaces). The module provides an ExternalDeviceManager interface used for configuring external TSS objects in chart and allows for ongoing updates to those objects.

5.8.1.10.13 TSSImportModuleProperties (Class)

This class holds all properties needed by the TSSImportModule including the name of the customized class used to acquire data from the external data source. Also, the name of the customized classes used to define the steps needed to translate the external TSS data into CHART TSS data.

5.8.1.10.14 TSSImportTranslationHandler (Class)

This class creates and handles TSSImportTranslationTasks. It inherits CommandQueue processing functionality from the EITaskHandler. It handles translation step classes that are specified in the props file and customized for the specific translation needed for the data provided by the external system. After translation the data is forwarded to the TSSImportHandler for import into CHART.

5.8.1.10.15UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.8.1.11 TSSImportTranslationClasses (Class Diagram)

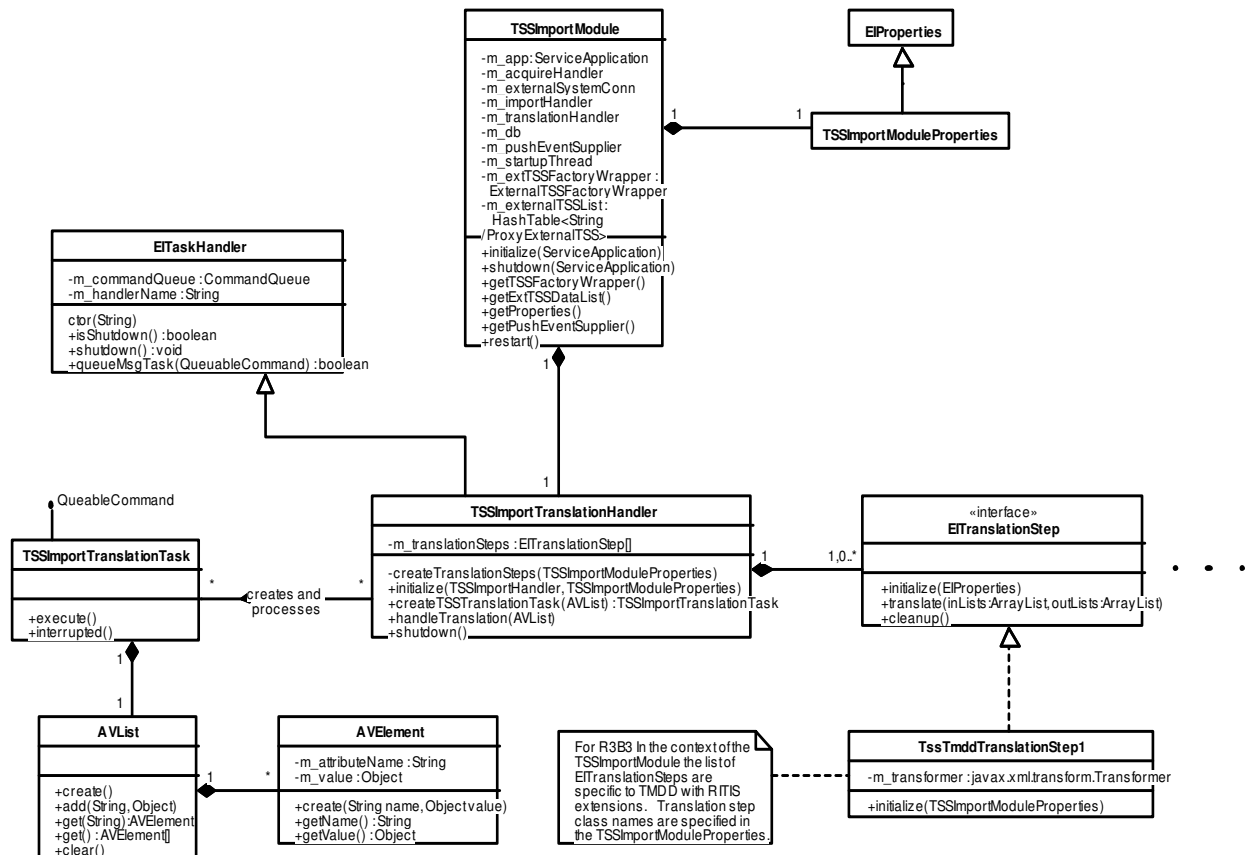


Figure 5-154 TSSImportTranslationClasses (Class Diagram)

5.8.1.11.1 AVElement (Class)

One element of an AVLList containing an attribute and a big-O-Object for storing/retrieving objects easily

5.8.1.11.2 AVLList (Class)

Generic collection of attribute and big-O-Object pairs used to move around the application.

5.8.1.11.3 EIProperties (Class)

This class supports properties that are generic to all External Interface modules such as log filenames.

5.8.1.11.4 EITaskHandler (Class)

This class is the base class for the EI Task Handlers. It manages a command queue that is used to process EI tasks (CommandQueueable) objects.

5.8.1.11.5 EITranslationStep (Class)

An EITranslationStep is the base class for all translations. Implementing translations define how to translate from a particular type of AVLlist to another particular type of AVLlist.

5.8.1.11.6 QueableCommand (Interface)

This interface is implemented by objects that can be placed on a command queue.

5.8.1.11.7 TSSImportModule (Class)

This module imports TSS data from an external source. The module allows for customizable (data source specific) data acquisition and translation using custom classes implementing generic interfaces). The module provides an ExternalDeviceManager interface used for configuring external TSS objects in chart and allows for ongoing updates to those objects.

5.8.1.11.8 TSSImportModuleProperties (Class)

This class holds all properties needed by the TSSImportModule including the name of the customized class used to acquire data from the external data source. Also, the name of the customized classes used to define the steps needed to translate the external TSS data into CHART TSS data.

5.8.1.11.9 TSSImportTranslationHandler (Class)

This class creates and handles TSSImportTranslationTasks. It inherits CommandQueue processing functionality from the EITaskHandler. It handles translation step classes that are specified in the props file and customized for the specific translation needed for the data provided by the external system. After translation the data is forwarded to the TSSImportHandler for import into CHART.

5.8.1.11.10 TSSImportTranslationTask (Class)

This class wraps an external TSS inventory or status message so it can be put on a command queue. When the command queue calls the execute() method, this class invokes the TSSImportTranslationHandler to translate it into CHART-centric TSS data. .

5.8.1.11.11 TssTmddTranslationStep1 (Class)

This class represents step one of a one step translation which will translate TMDD TSS inventory and TSS status messages to CHART-centric terms. An example of when a multiple step translation may be needed is when a complex message needs to be simplified

before further translation can be done. Multiple simple steps may be able to accomplish what one complex step could. For ease of maintenance, multiple steps may be better in that case.

5.8.2 Sequence Diagrams

5.8.2.1 DMSImportAcquireTask:execute (Sequence Diagram)

This diagram depicts the behavior of the execute() method of the DMSImportAcquireTask. It is called when this task reaches the head of the Acquire command queue. This method is responsible for breaking the external message into individual DMS messages (inventory and status). A special case it handles is that it must place a marker in the message stream after the complete set of refresh events are received after a reconnect with RITIS. After the Translation Handler passes this marker to the DMS Import handler, the DMS Import handler knows that all external DMS objects have recently been refreshed and it should close all events whose last update time is older than the time in the marker message.

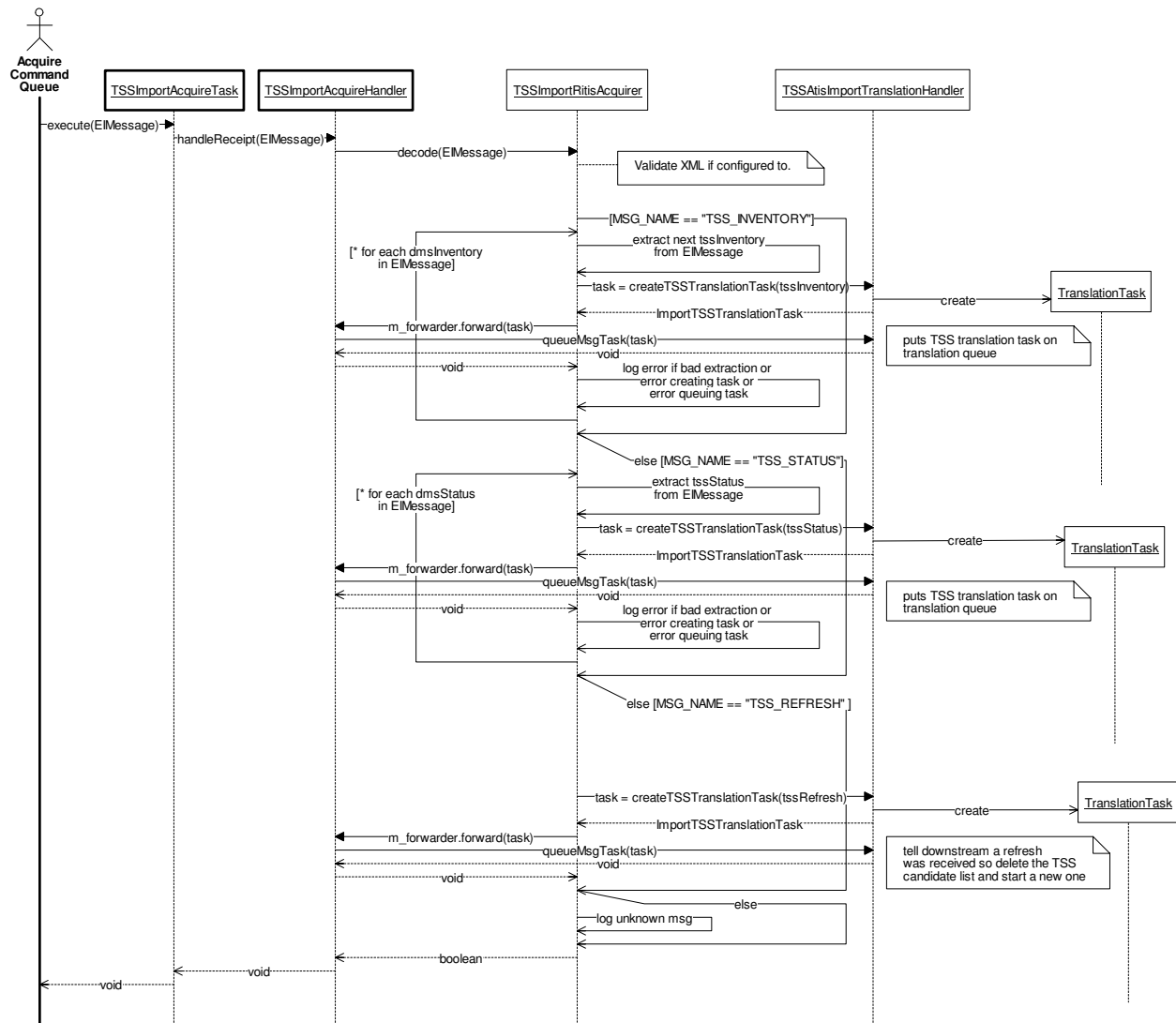


Figure 5-155 DMSImportAcquireTask:execute (Sequence Diagram)

5.8.2.2 DMSImportHandler:getCandidates (Sequence Diagram)

The DMSImportHandler.getCandidates() method is part of the ExternalDeviceCandidateSupporter interface and is called by the ExternalDeviceManagerImpl. The collection of ExternalDMSCandidates is traversed to create a list of ExternalDeviceCandidate objects used by the ExternalDeviceManagerImpl. If the candidate is in the list of ExternalDMS objects known to Chart, the candidate's INCLUDED flag is set. If the candidate is in the excluded list, the candidate's EXCLUDED flag is set. Note: if the candidate is in both lists, we assume the Include list is correct and remove it from the excluded list. Only one flag should be set. The correctly populated list of ExternalDeviceCandidates is then returned to the caller.

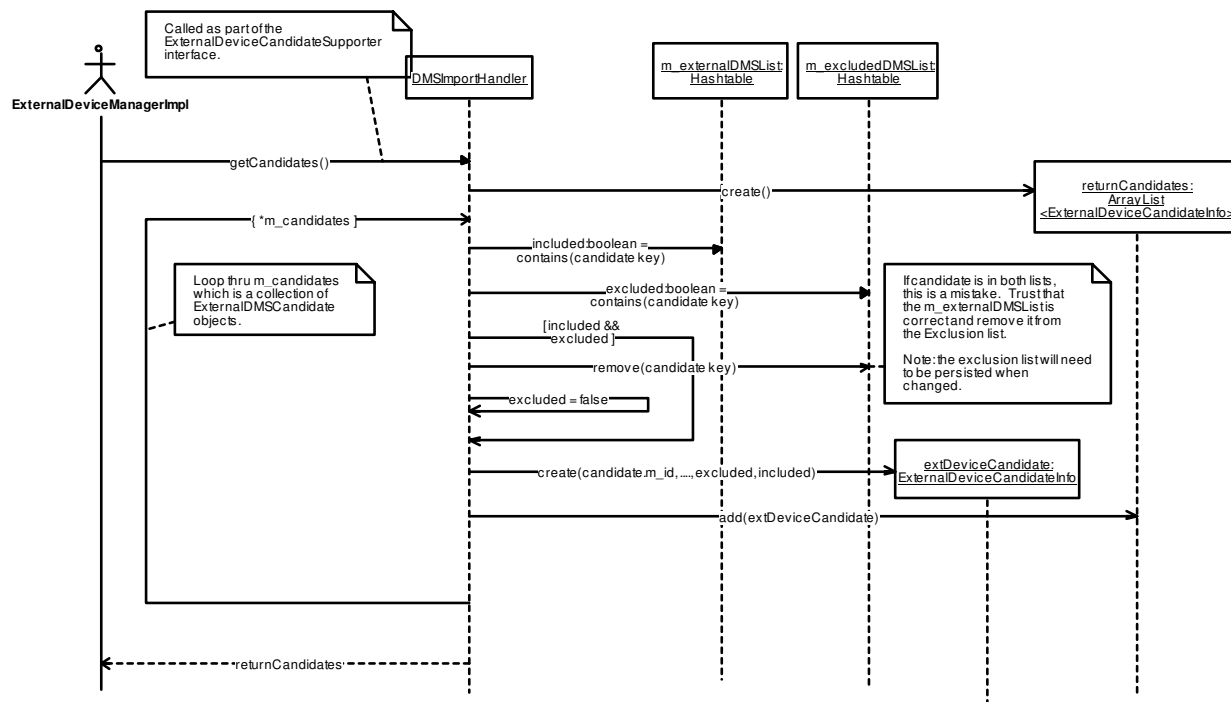


Figure 5-156 DMSImportHandler:getCandidates (Sequence Diagram)

5.8.2.3 DMSImportRitisAcquirer:connectIfNecessary (Sequence Diagram)

This diagram depicts the connectIfNecessary() method of the DMSImportRitisAcquirer class. This method is called at initiation and periodically after that. When invoked, if the RITIS connection is failed or stale (no activity on the connection for while - see properties), it attempts a reconnection. Its primary duty is to rebuild the JMS connection with RITIS by registering to have the OnMessage() method called whenever RITIS has a message to send. It is expected that the stale connection value will be large enough to not present a performance problem for either CHART or RITIS but small enough to be responsive to users if the connection is temporarily lost. The RITIS connection status is updated if a reconnect is attempted.

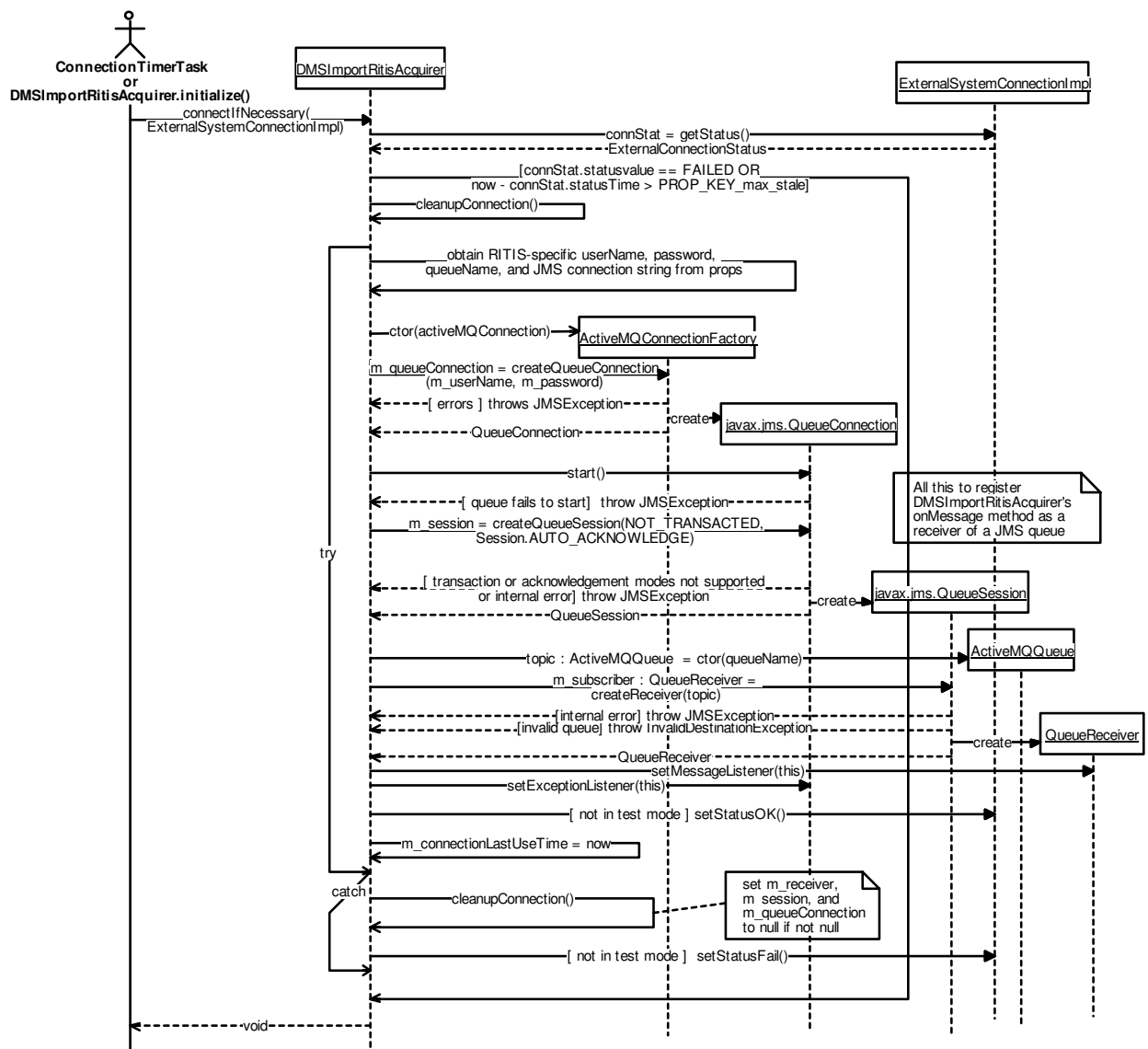


Figure 5-157 DMSImportRitisAcquirer:connectIfNecessary (Sequence Diagram)

5.8.2.4 DMSImportRitisAcquirer:initialize (Sequence Diagram)

This diagram depicts the initialize() method of the DMSImportRitisAcquirer class. After attempting to connect to RITIS and updating the connection status, it kicks off a periodic task that checks the connection status (see connectIfNecessary()) and reconnects, if necessary.

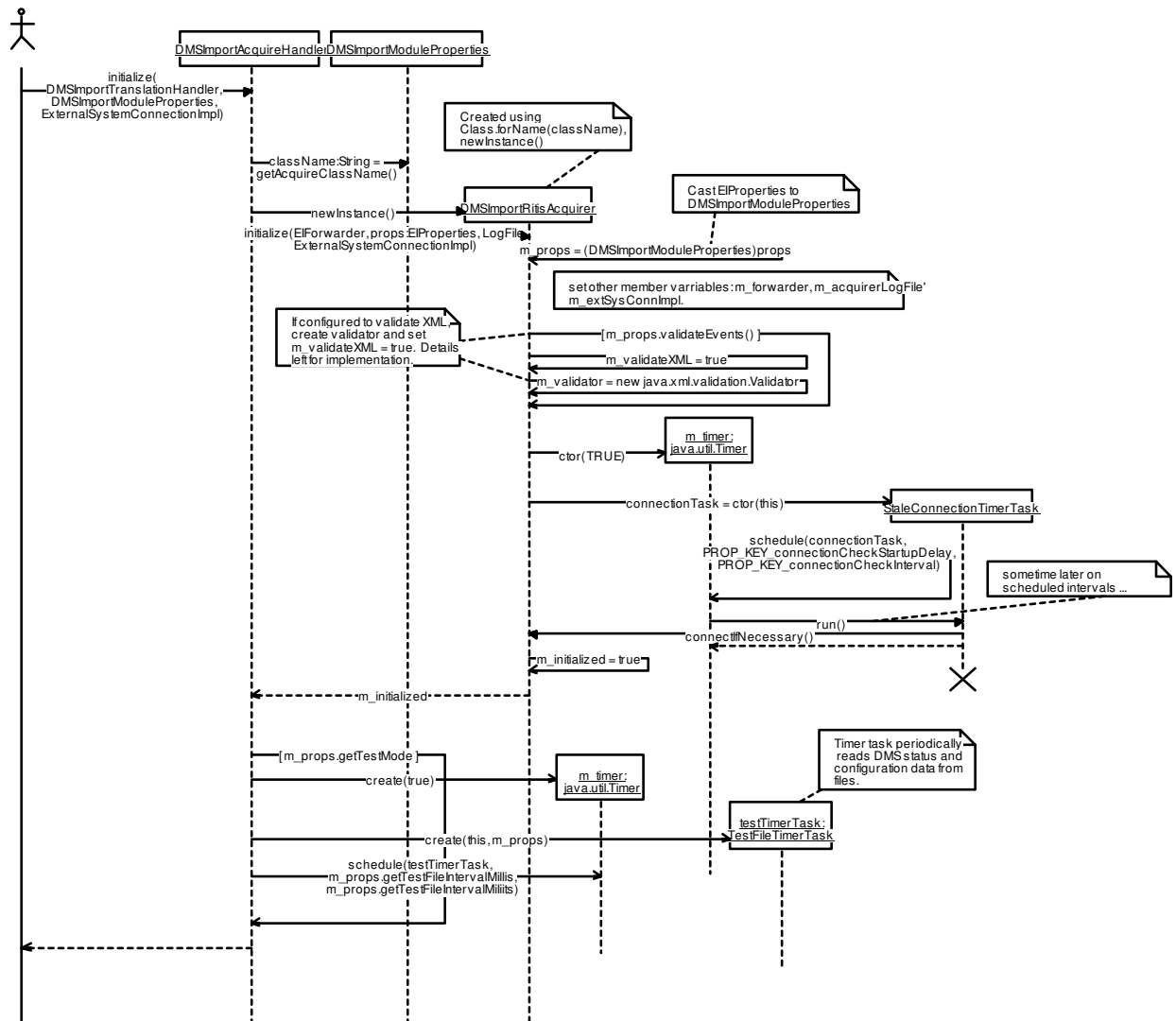


Figure 5-158 DMSImportRitisAcquirer:initialize (Sequence Diagram)

5.8.2.5 DMSImportRitisAcquirer:onMessage (Sequence Diagram)

This diagram depicts the onMessage() method of the DMSImportRitisAcquirer class. This method is called when RITIS has a DMS inventory or status message for CHART. To ensure CHART is responsive to RITIS, the onMessage() method's only job is to create an Acquire task containing the external message and put it on the Acquire command queue in the proper format and be ready for the next message from RITIS.

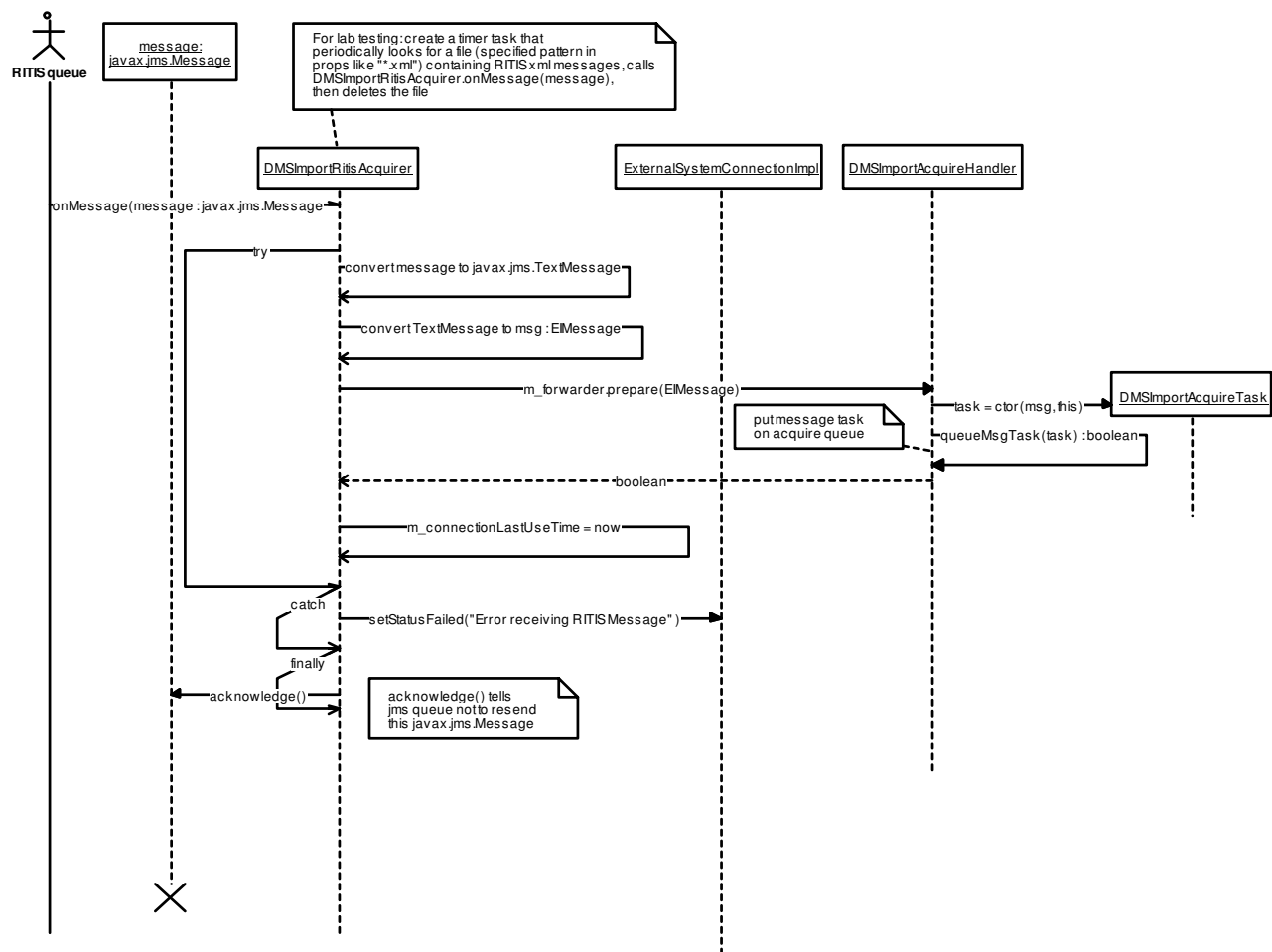


Figure 5-159 DMSImportRitisAcquirer:onMessage (Sequence Diagram)

5.8.2.6 EventImportModule:ExtSysConnStatusUpdate (Sequence Diagram)

This diagram depicts the setStatus() method for the ExternalSystemConnectionImpl. If the state changes a CORBA event is pushed with the current status. An event is pushed on any update, if configured to do so.

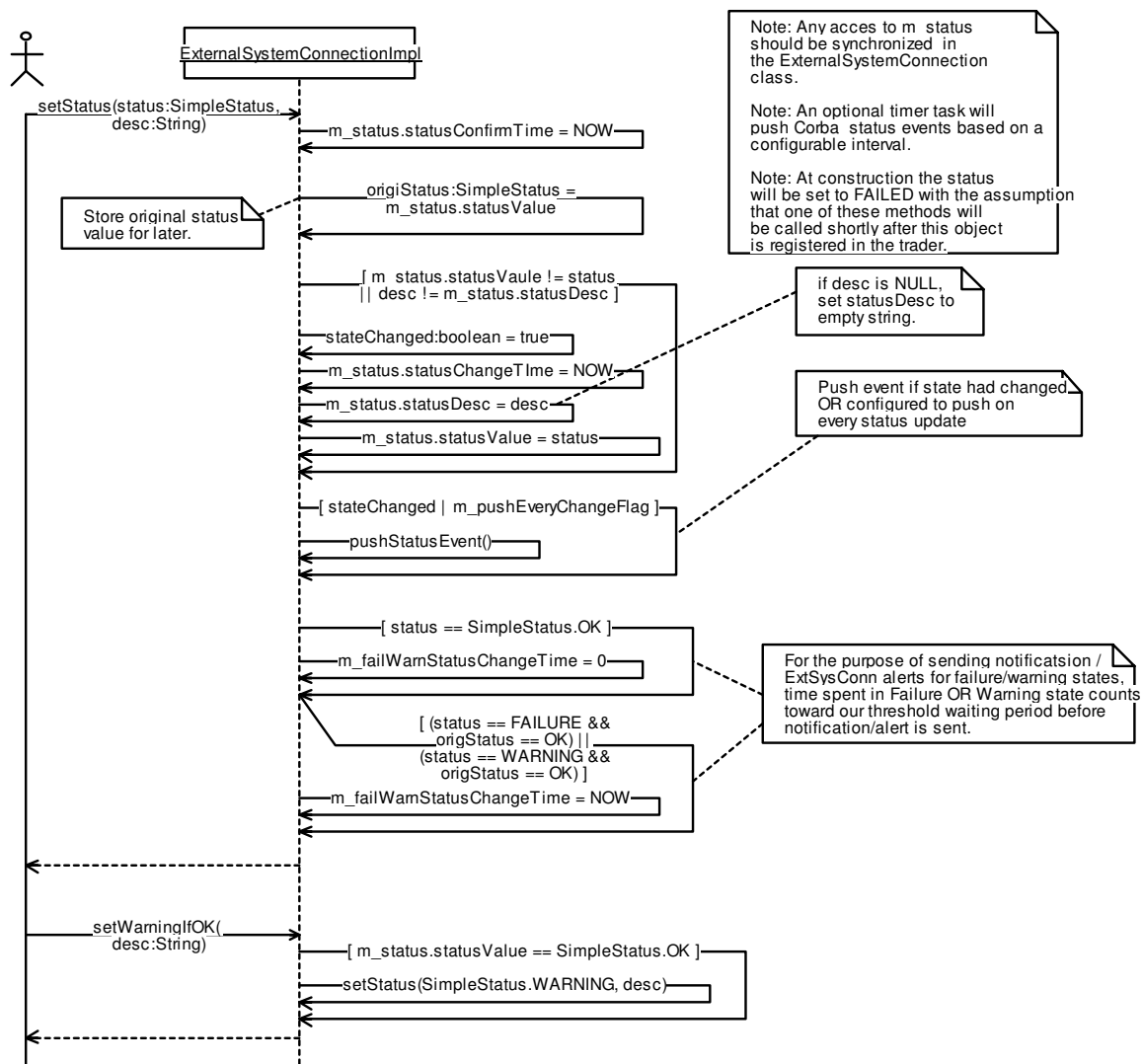


Figure 5-160 EventImportModule:ExtSysConnStatusUpdate (Sequence Diagram)

5.8.2.7 EventImportRitisAcquirer:connectIfNecessary (Sequence Diagram)

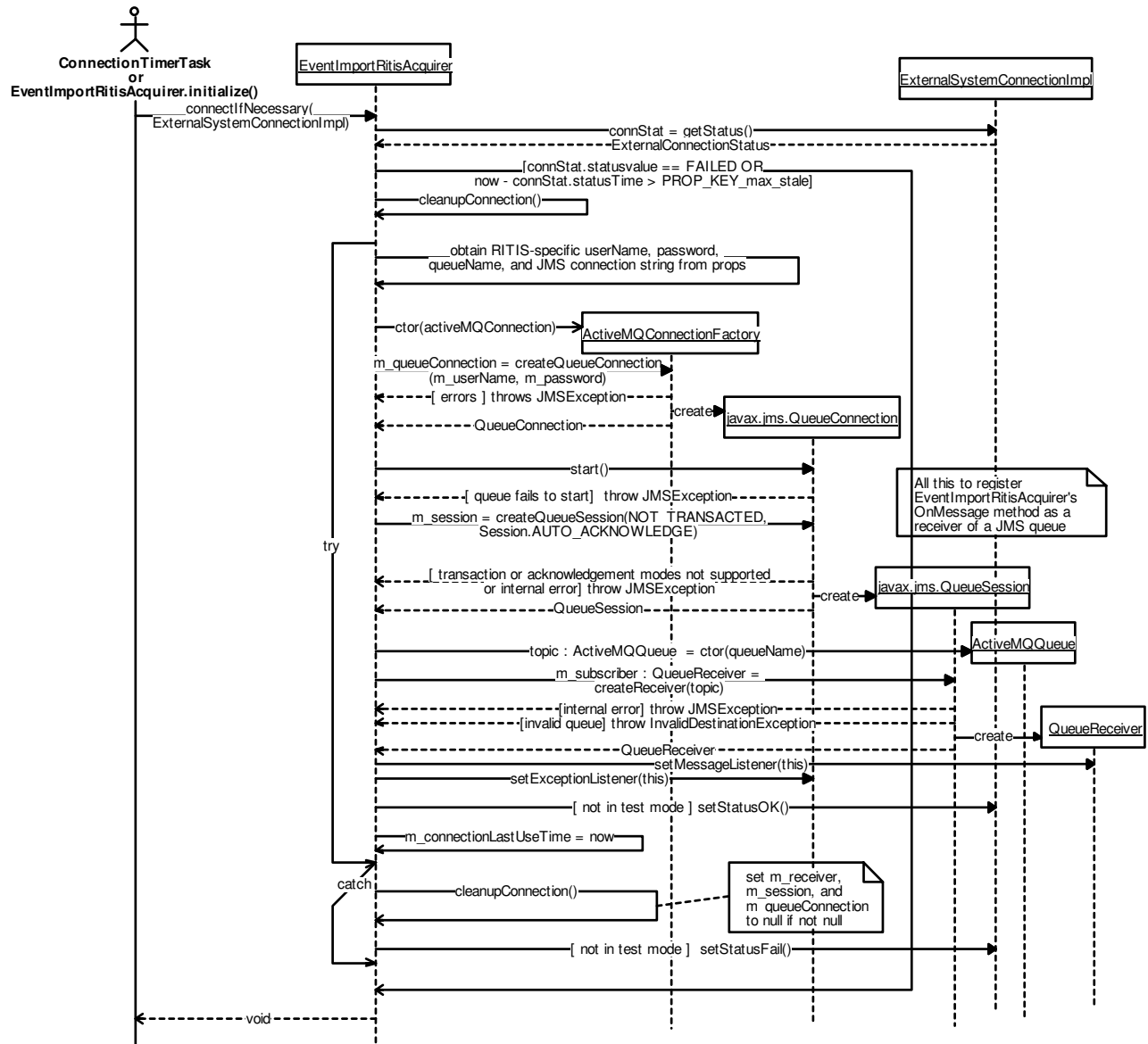


Figure 5-161 EventImportRitisAcquirer:connectIfNecessary (Sequence Diagram)

5.8.2.8 ExternalDeviceManagerImpl:searchCandidates (Sequence Diagram)

This sequence diagram depicts the processing involved when a client requests a search of external device candidates. The ExternalDeviceManagerImpl utilized a ExternalDeviceCandidateSupporter to retrieve a list of ExternalDeviceCandidates. The ExternalDeviceCandidateSupporter interface is implemented by either the DMSImportModule's DMSImportHandler or the TSSImportModule's TSSImportHandler. If the search criteria passed in from the client contains GeoAreas to search, a list of Polygons representing the GeoAreas to be searched is created. The search criteria is applied to the candidates in order to build a list of candidates meeting the search criteria. This result is returned to the client after being wrapped in a CandidateGroup object.

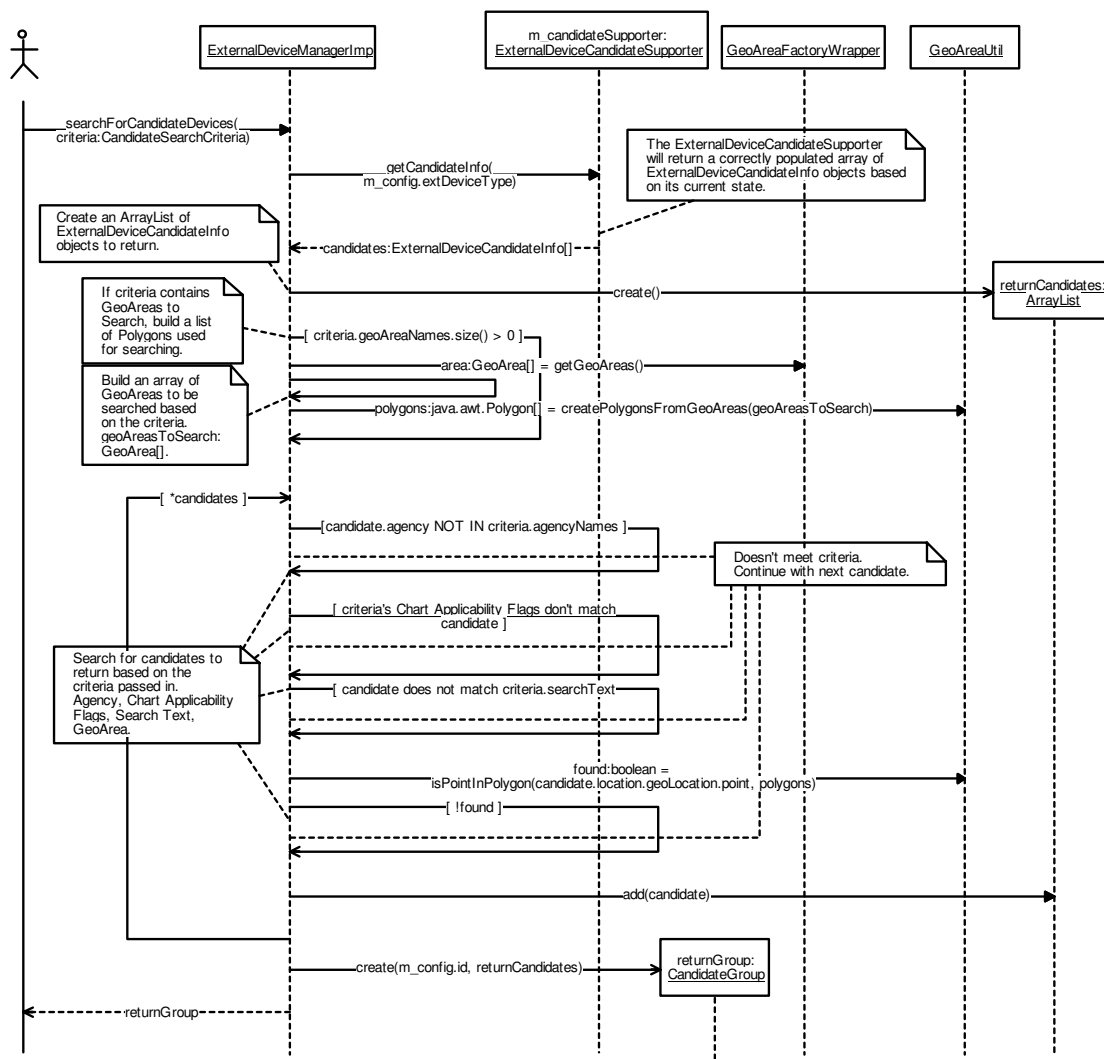


Figure 5-162 ExternalDeviceManagerImpl:searchCandidates (Sequence Diagram)

5.8.2.9 ExternalDeviceManagerImpl:setCandidates (Sequence Diagram)

The sequence diagram depicts the ExternalDeviceManagerImpl.setCandidates() method. This method is called by the gui to apply directives for configuring External Devices. The method will validate the data passed from the gui and throw an exception if invalid data is found. After validation the ExternalDeviceManagerImpl will call the ExternalDeviceSupporter interface (implemented by the DMSImportHandler in this diagram) to set the external device candidate directives. For each ExternalDeviceCandidateInfo object passed in, follow the directives in each object to maintain the lists of include ExternalDMS objects as well as excluded ext devices.

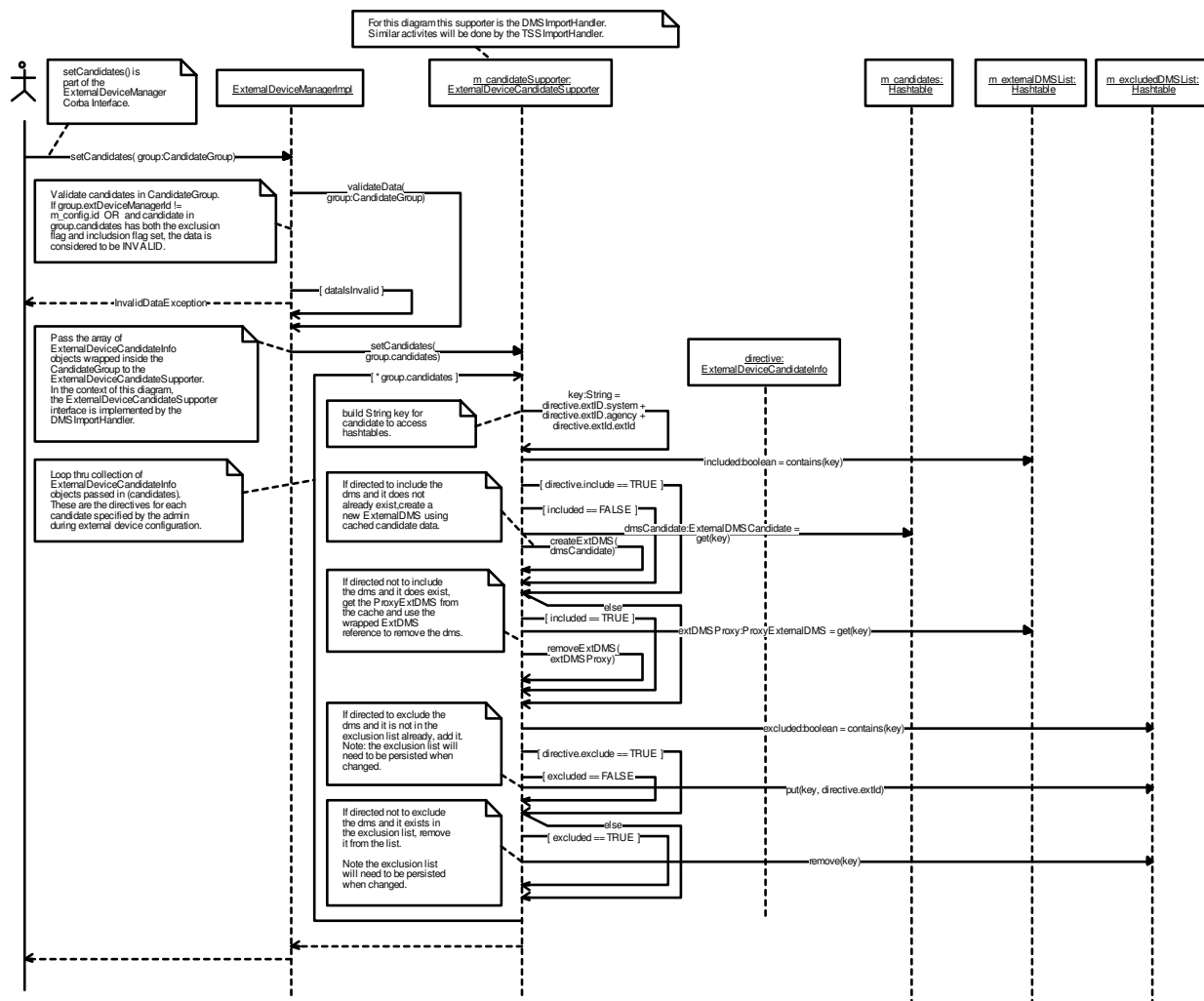
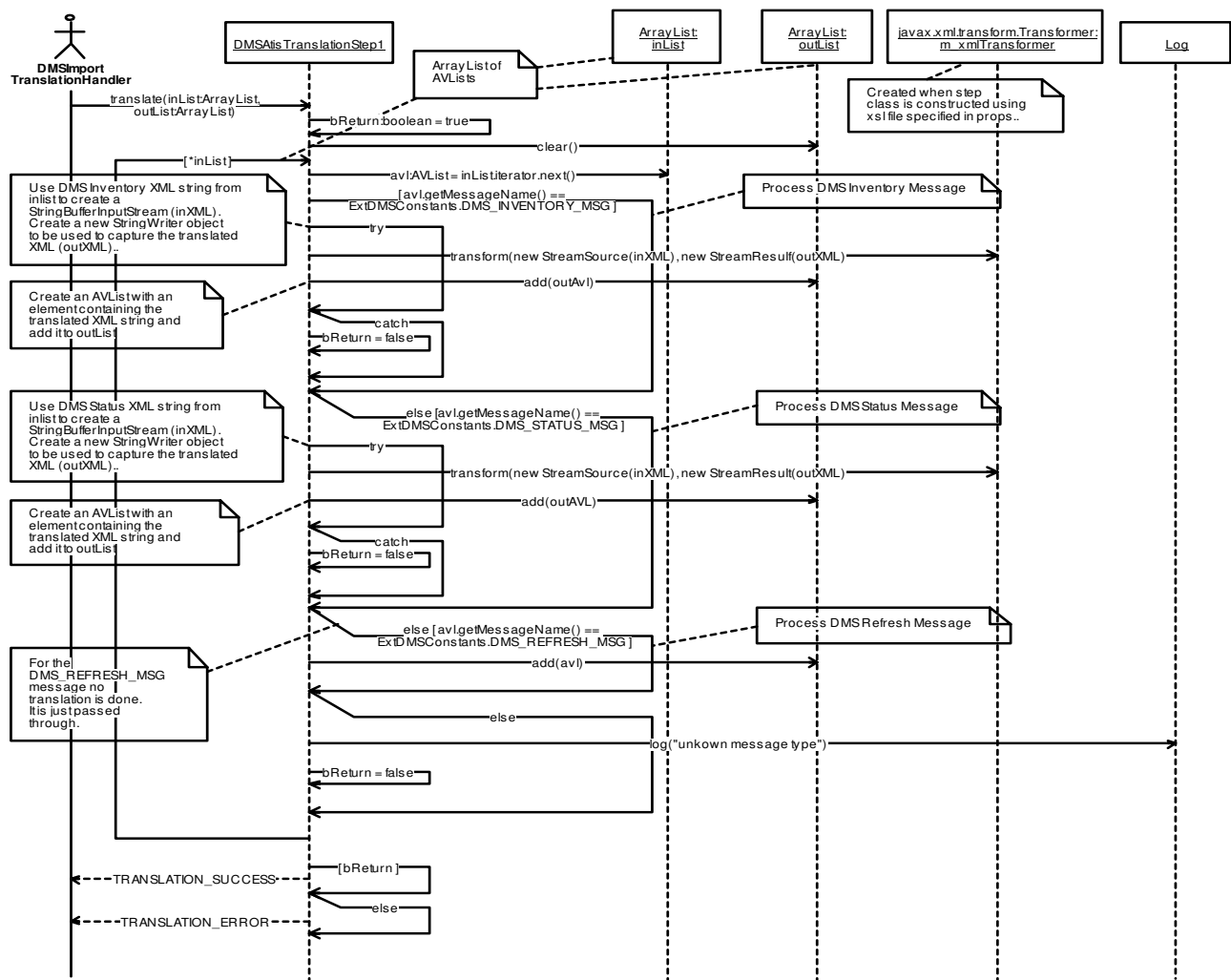


Figure 5-163 ExternalDeviceManagerImpl:setCandidates (Sequence Diagram)

5.8.2.10 ExternalInterfaceModule:dmsTranslationStep1Translate (Sequence Diagram)

This diagram depicts the translation of external event messages into Chart Event data. The `DMSAtisTranslationStep1.translate()` method is called by the `DMSImportTranslationHandler.handleTranslation()` method. It takes in a `AVList`

and based on the message name processes it accordingly. For EVENT related messages it pulls the XML string from the first element in the `AVList`. This xml string contains information for one specific event or incident. It transforms the XML into a CHART-specific XML string using XSL Transformation. The resulting XML string is then added to the outbound `AVList` and the method returns. For REFRESH messages no translation is done. The message is just added to the outbound `AVList` and the method returns.



5.8.2.11 ExternalInterfaceModule:handleDITranslationTask (Sequence Diagram)

This diagram depicts how each DMS message is iterated on as it is translated from the external XML format into the internal format. The result is put on the DMSImportHandler's command queue for use as a candidate object and possibly to update the status or configuration of an external DMS that has already been imported.

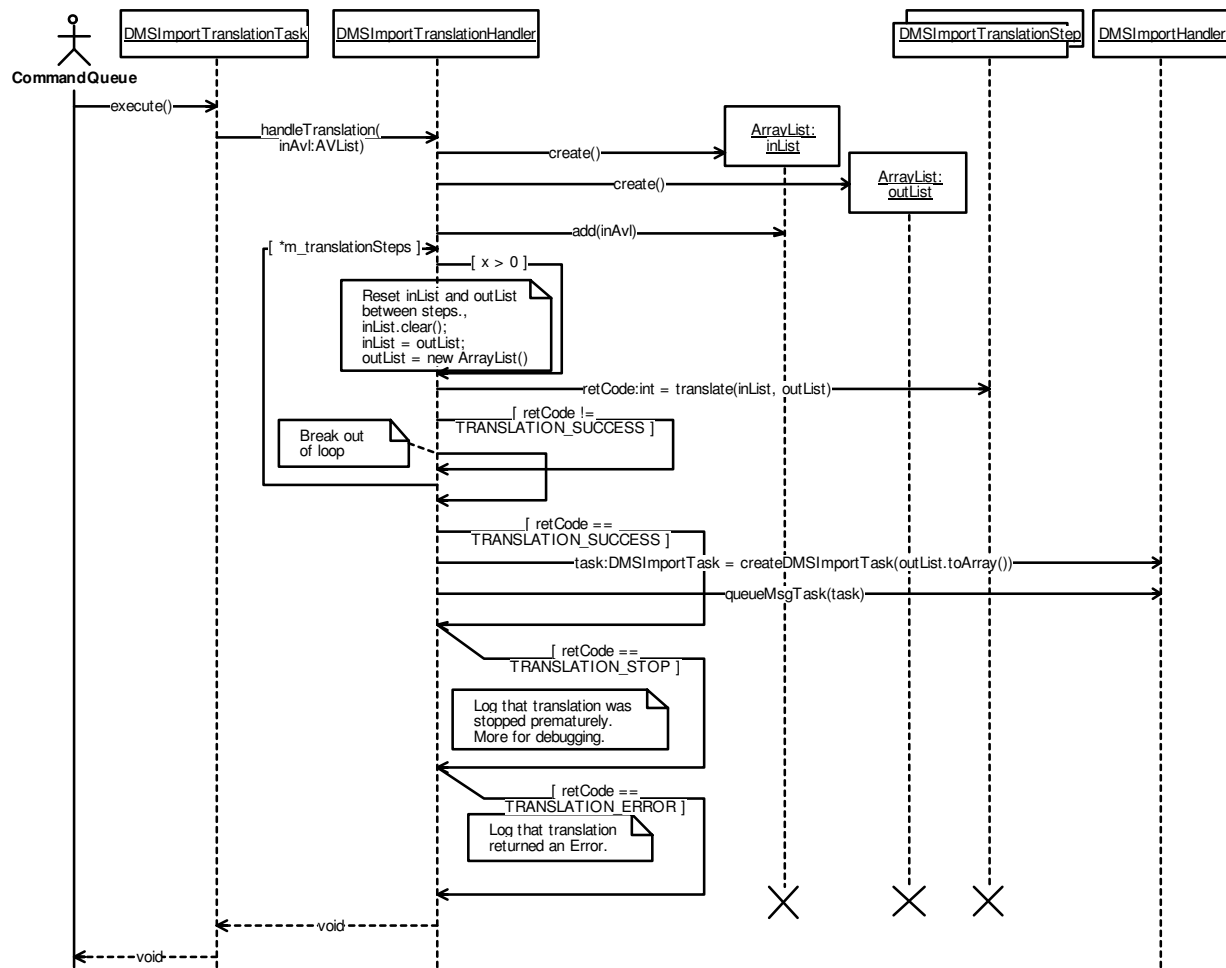


Figure 5-165 ExternalInterfaceModule:handleDITranslationTask (Sequence Diagram)

5.8.2.12 ExternalInterfaceModule:handleDMSImportTask (Sequence Diagram)

This diagram depicts how the translated DMS inventory and status messages are used to update an external candidate DMS (in case an administrator wishes to import that object) and to update the inventory or status of an external DMS that has already been imported.

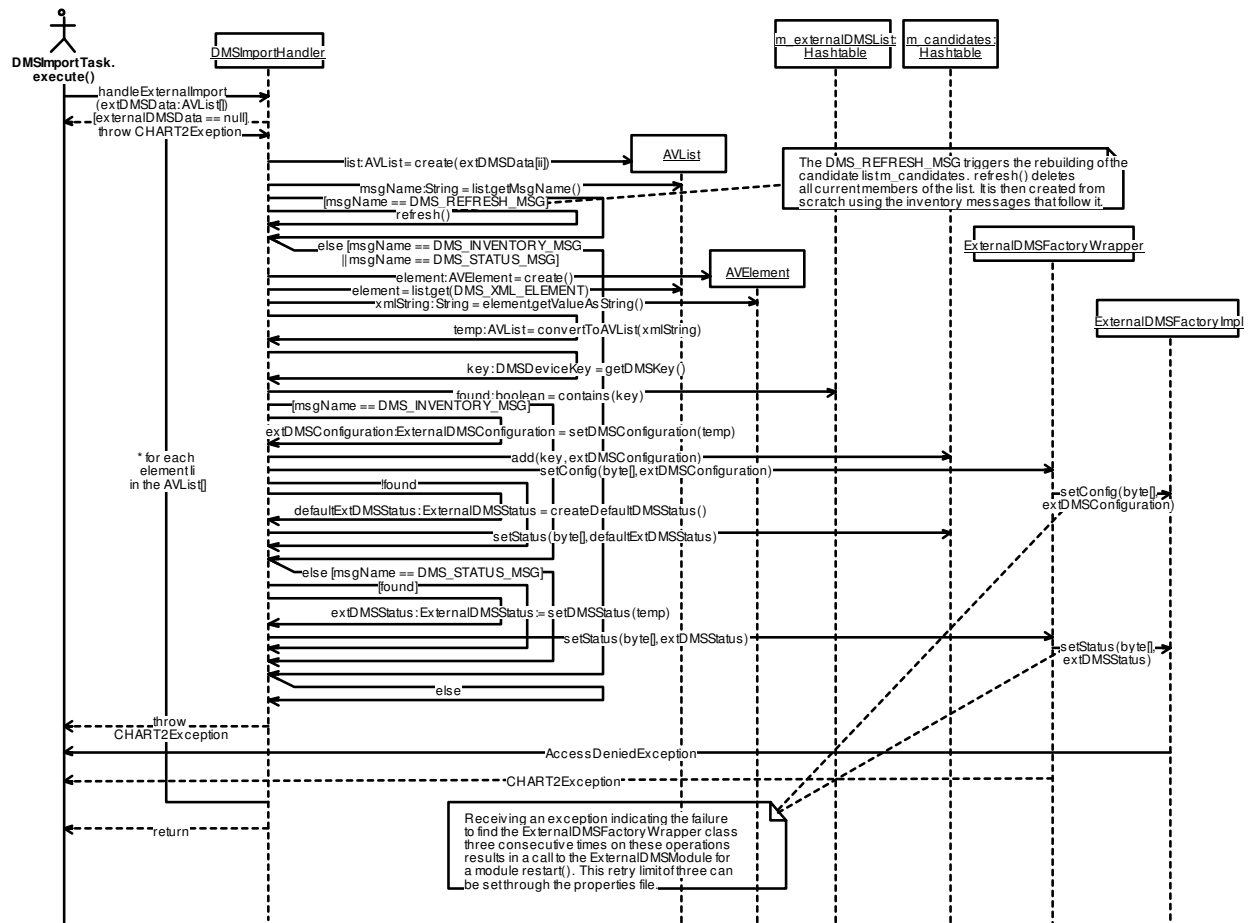


Figure 5-166 ExternalInterfaceModule:handleDMSImportTask (Sequence Diagram)

5.8.2.13 ExternalInterfaceModule:handleExternalImport (Sequence Diagram)

This diagram shows how the EventImportHandler class imports external events. After copying the XML from the translation handler into an AVLList, it validates the data. If invalid it logs that it is ignoring the event. If valid, it populates the BasicEventData structure. It then looks for the external event in the ObjectCache based on the agency identification and event Identification. If the event exists it updates the event using the TrafficEventFactoryWrapper. If the event is new, it calls TrafficEventFactoryWrapper's createExternalTrafficEvent() method to add the new event to the factory which, in turn, updates the Object Cache.

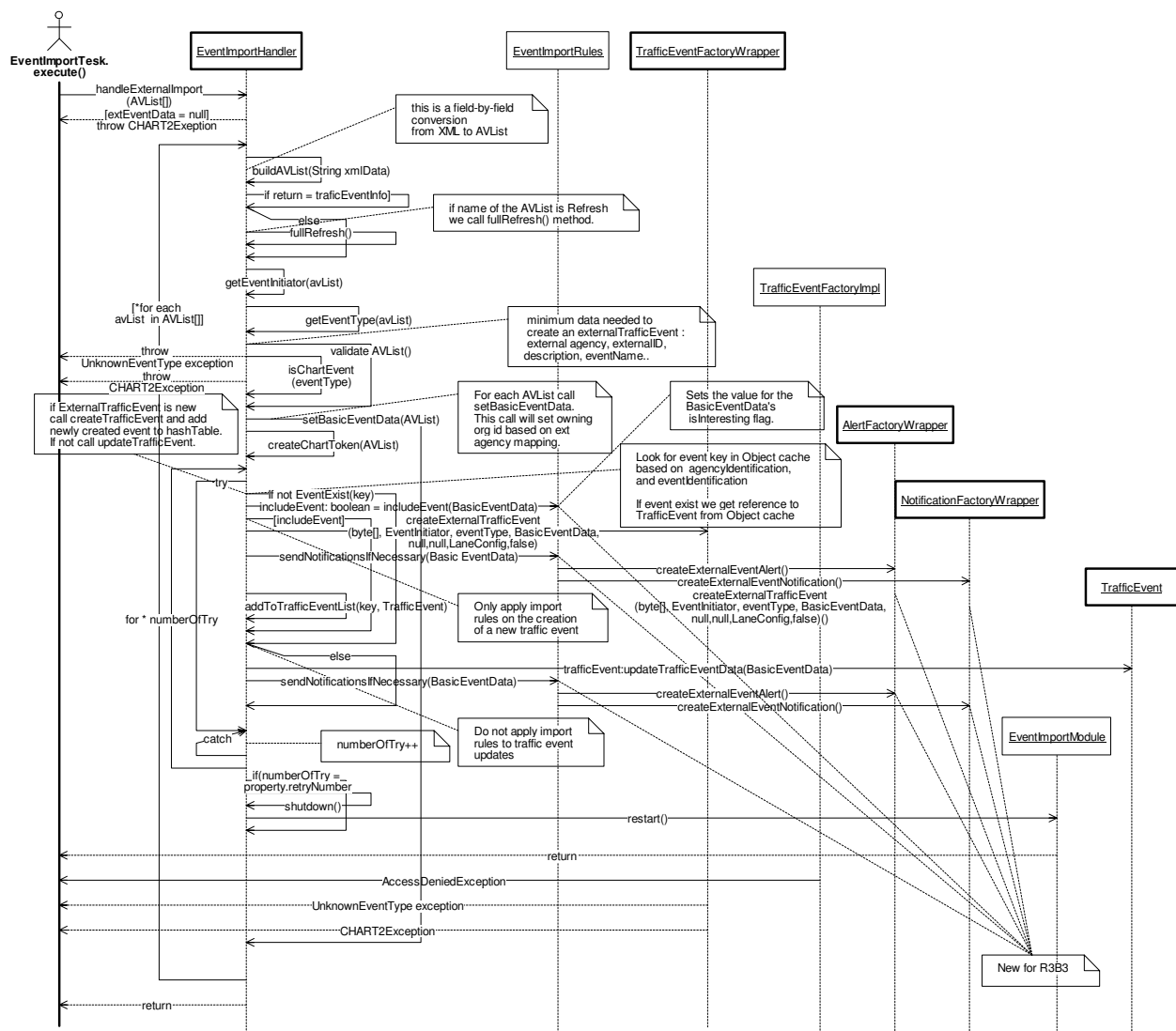


Figure 5-167 ExternalInterfaceModule:handleExternalImport (Sequence Diagram)

5.8.2.14 ExternalInterfaceModule:handleTITranslationTask (Sequence Diagram)

This diagram depicts how each TSS message is iterated on as it is translated from the external XML format into the internal format. The result is put on the TSSImportHandler's command queue for use as a candidate object and possibly to update the status or configuration of an external TSS that has already been imported.

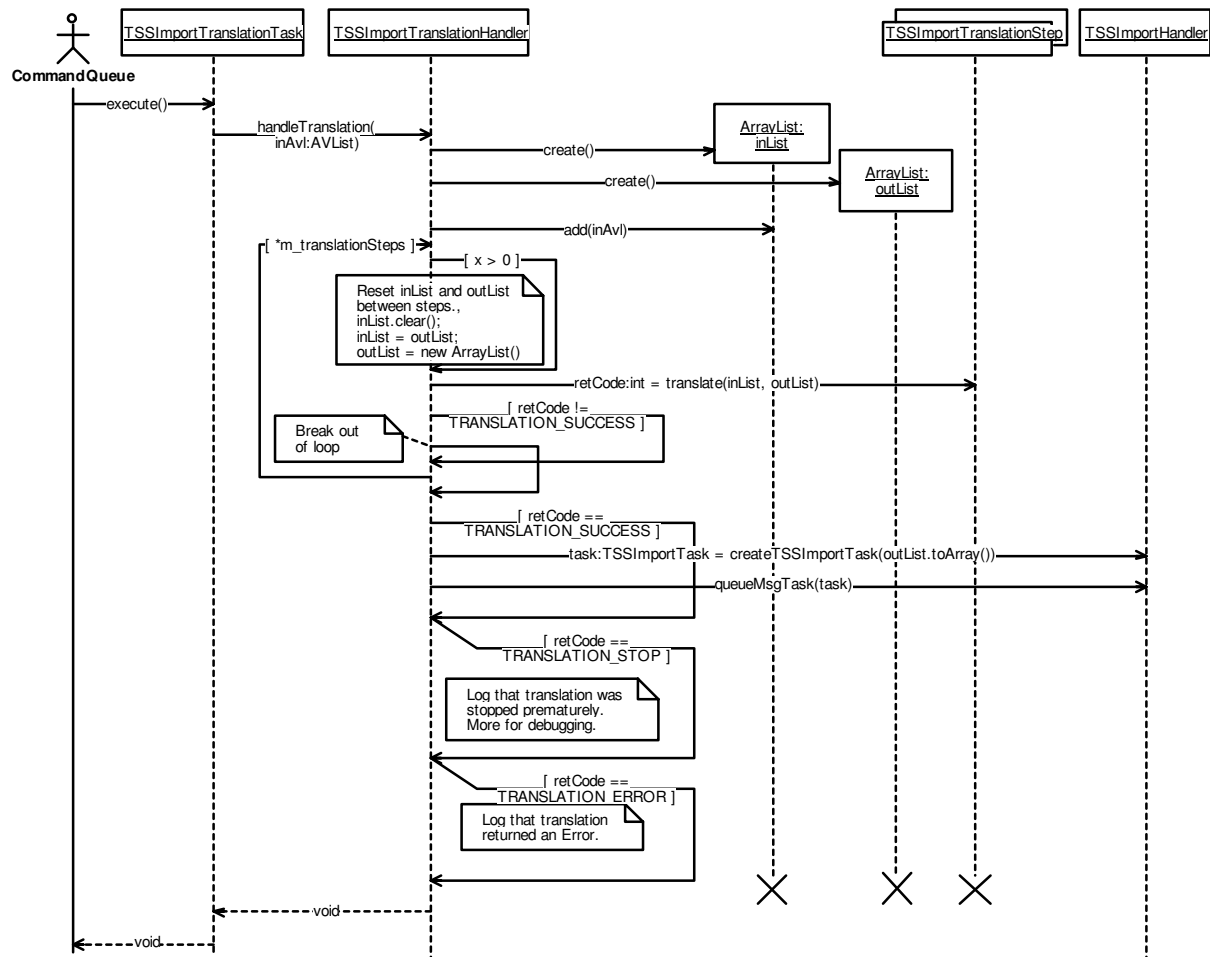


Figure 5-168 ExternalInterfaceModule:handleTITranslationTask (Sequence Diagram)

5.8.2.15 ExternalInterfaceModule:handleTSSImportTask (Sequence Diagram)

This diagram depicts how the translated TSS inventory and status messages are used to update an external candidate TSS (in case an administrator wishes to import that object) and to update the inventory or status of an external TSS that has already been imported.

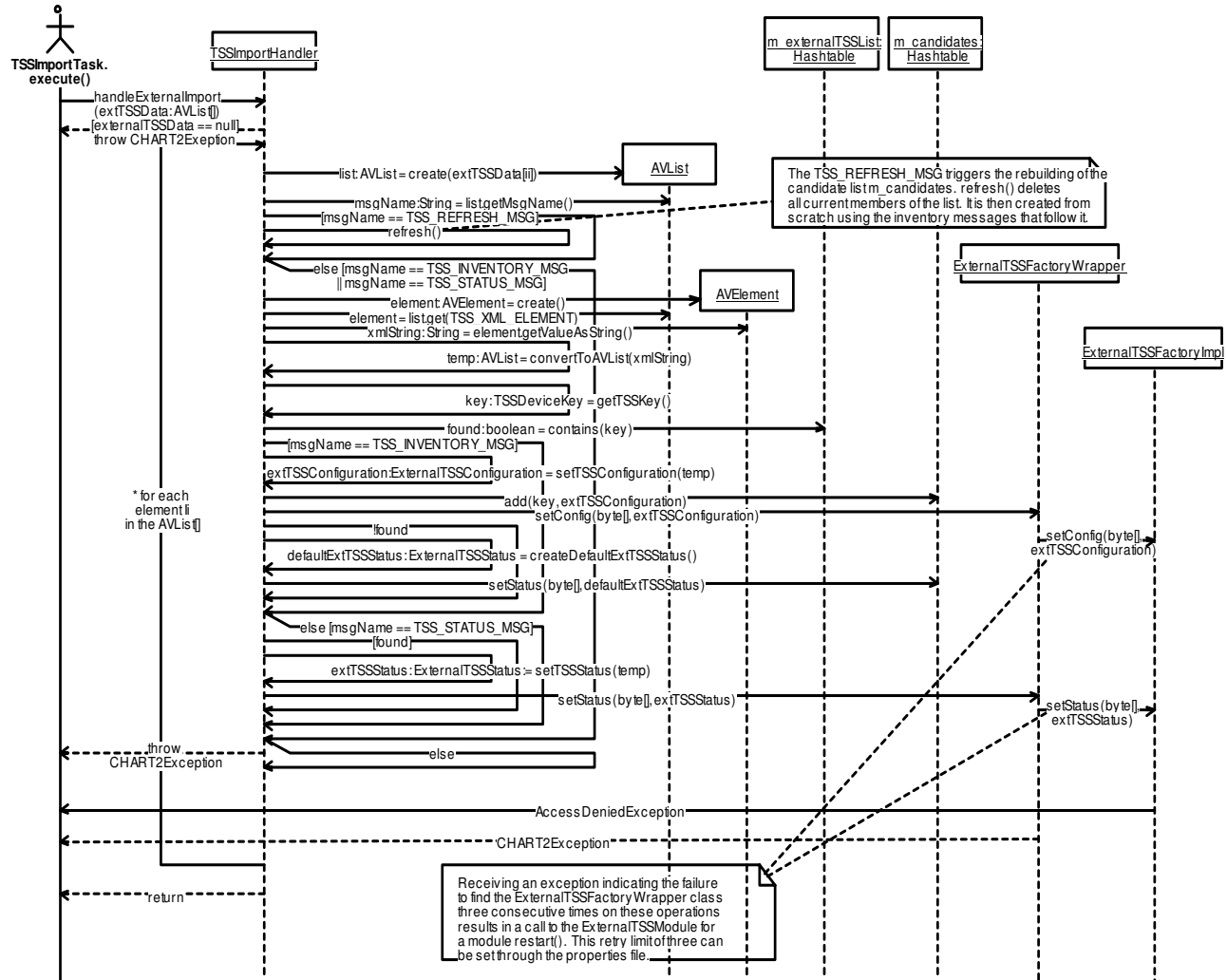


Figure 5-169 ExternalInterfaceModule:handleTSSImportTask (Sequence Diagram)

5.8.2.16 ExternalInterfaceModule:initializeDMSImportModule (Sequence Diagram)

This diagram depicts the initialization of the DMSImportModule. After the props file is read the ExternalSystemInterface is created and activated with the ORB. The Handlers are created and then initialized in this order: DMSImportHandler, DMSImportTranslationHandler, DMSImportAcquireHandler. Note that the DMSImportHandler may wait forever looking for a valid DMSFactory. This is intentional because without the factory, the other handlers have nothing to do.

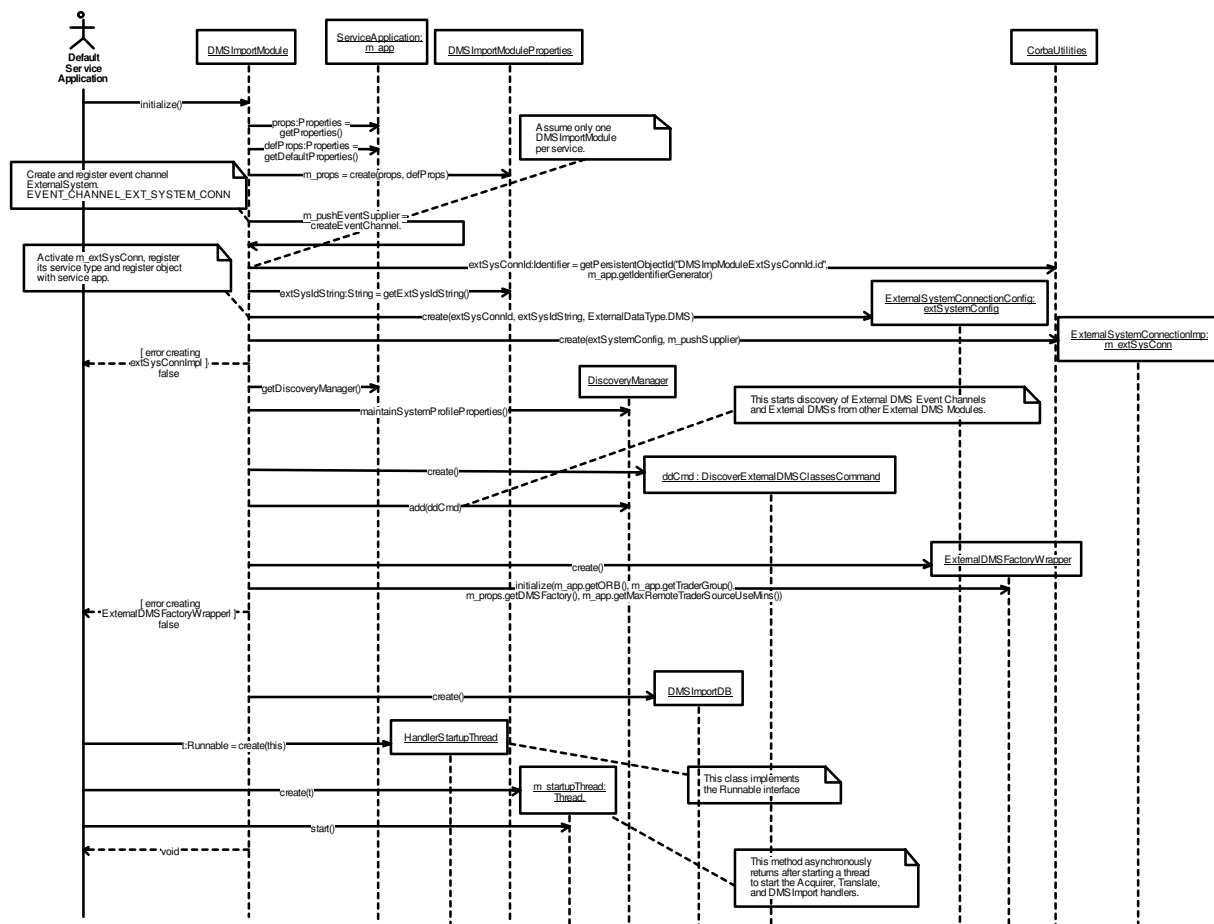


Figure 5-170 ExternalInterfaceModule:initializeDMSImportModule (Sequence Diagram)

5.8.2.17 ExternalInterfaceModule:initializeEventImportModule (Sequence Diagram)

This diagram depicts the initialization of the EventImportModule. After the props file is read the ExternalSystemInterface is created and activated with the ORB. The Handlers are created and then initialized in this order: EventImportHandler, EventImportTranslationHandler, EventImportAcquireHandler. Note that the EventImportHandler may wait forever looking for a valid TrafficEventFactory. This is intentional because without the factory, the other handlers have nothing to do.

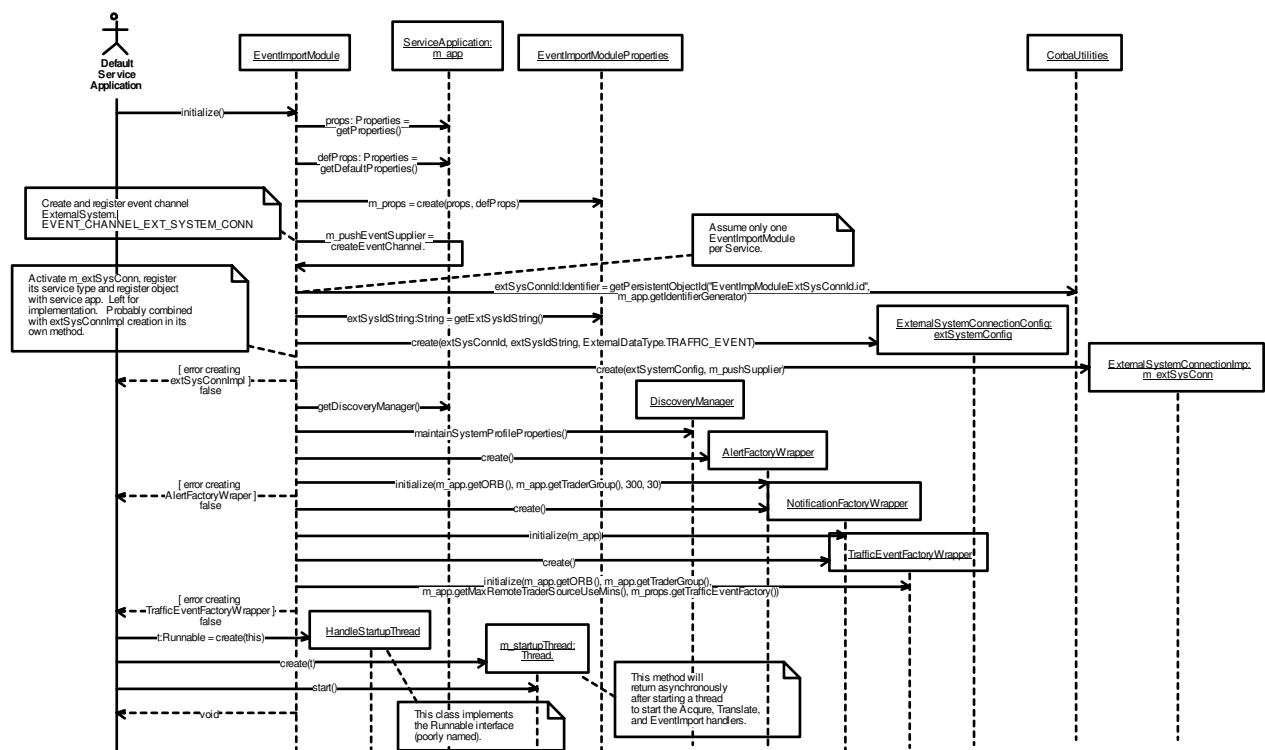


Figure 5-171 ExternalInterfaceModule:initializeEventImportModule (Sequence Diagram)

5.8.2.18 ExternalInterfaceModule:initializeTSSImportModule (Sequence Diagram)

This diagram depicts the initialization of the TSSImportModule. After the props file is read the ExternalSystemInterface is created and activated with the ORB. The Handlers are created and then initialized in this order: TSSImportHandler, TSSImportTranslationHandler, TSSImportAcquireHandler. Note that the TSSImportHandler may wait forever looking for a valid TSSFactory. This is intentional because without the factory, the other handlers have nothing to do.

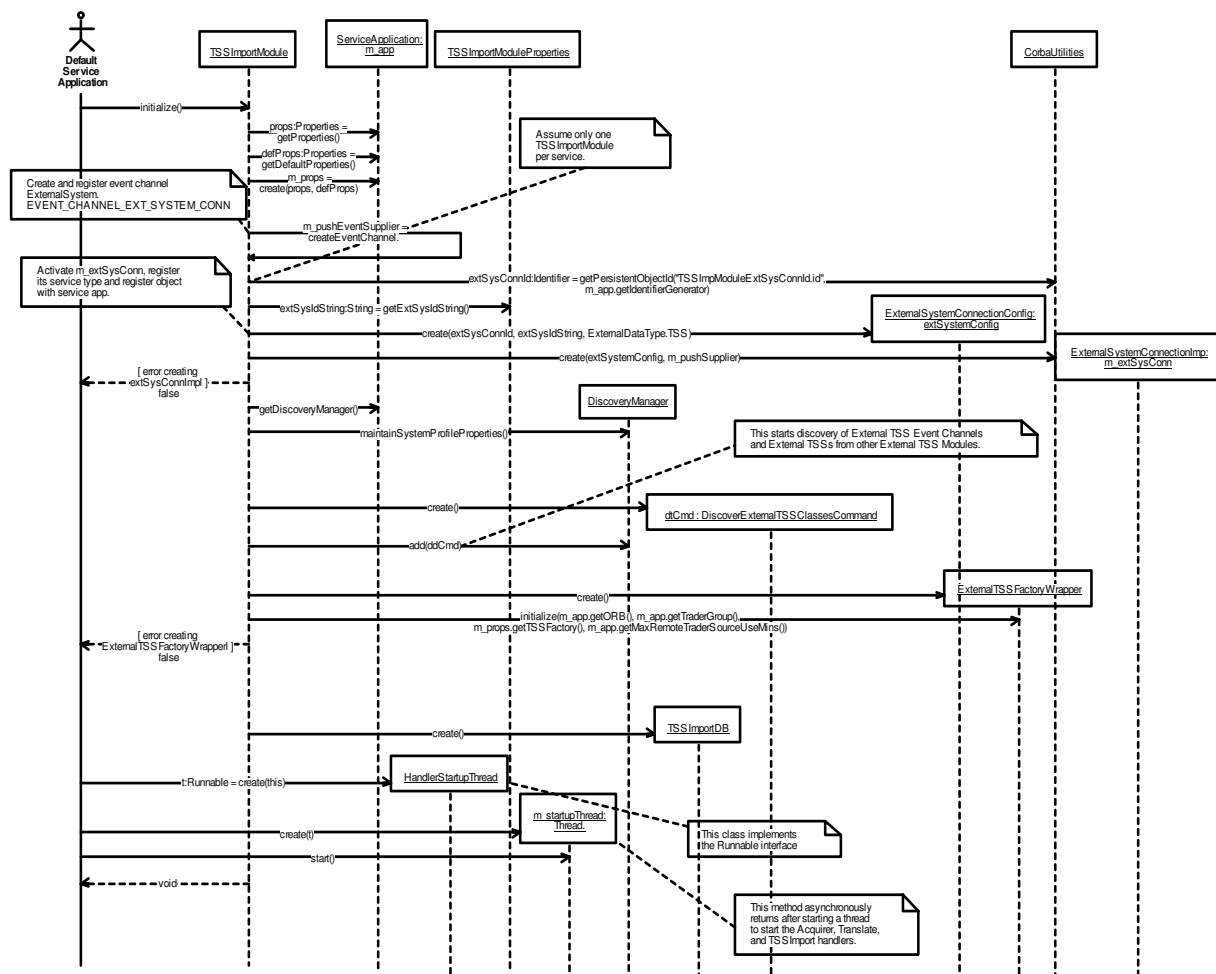


Figure 5-172 ExternalInterfaceModule:initializeTSSImportModule (Sequence Diagram)

5.8.2.19 ExternalInterfaceModule:restartDMSImportModule (Sequence Diagram)

This diagram depicts the restart process of the DMSImportModule. It begins with the creation of a restart task that manages the restart process independent of the calling method. It is important to point out that the module itself does not restart as the method name might imply. Rather, the three event import data handlers, the DMSImportHandler, the DMSSImportTranslationHandler, and the DMSImportAcquireHandler, are each shutdown and restarted with the intent of purging and resetting the data import queue. Order matters and the handlers are stopped in the reverse order from their original startup. During the time in which the data handlers are shutdown, the external connection status is forced to "failed" to bring about a connection reset with the external data source. Once this is complete, the restart task creates the event import data handlers anew. All three handlers being successfully created, they are, in turn, initialized so that they begin the external event import processing.

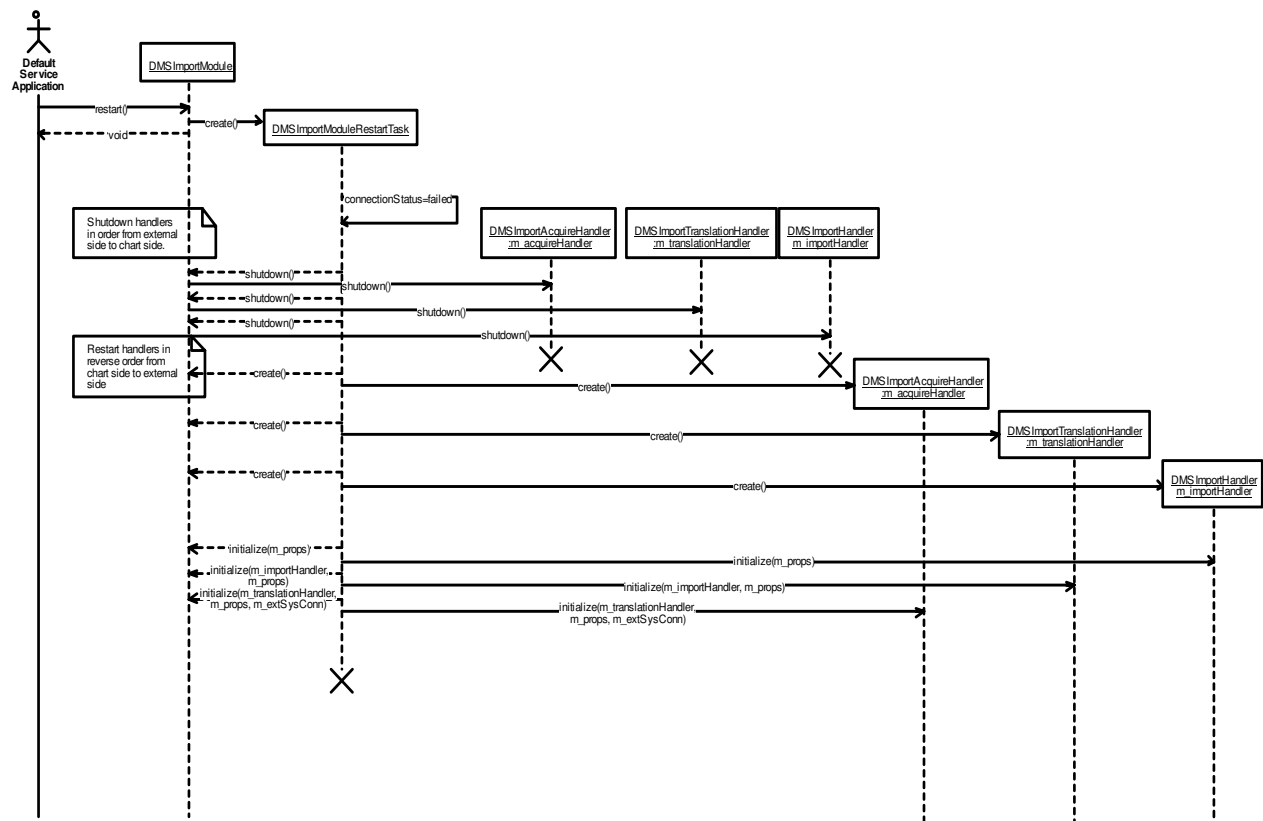


Figure 5-173 ExternalInterfaceModule:restartDMSImportModule (Sequence Diagram)

5.8.2.20 ExternalInterfaceModule:restartDMSImportModule (Sequence Diagram)

This diagram depicts the restart process of the TSSImportModule. It begins with the creation of a restart task that manages the restart process independent of the calling method. It is important to point out that the module itself does not restart as the method name might imply. Rather, the three event import data handlers, the TSSImportHandler, the TSSImportTranslationHandler, and the TSSImportAcquireHandler, are each shutdown and restarted with the intent of purging and resetting the data import queue. Order matters and the handlers are stopped in the reverse order from their original startup. During the time in which the data handlers are shutdown, the external connection status is forced to "failed" to bring about a connection reset with the external data source. Once this is complete, the restart task creates the event import data handlers anew. All three handlers being successfully created, they are, in turn, initialized so that they begin the external event import processing.

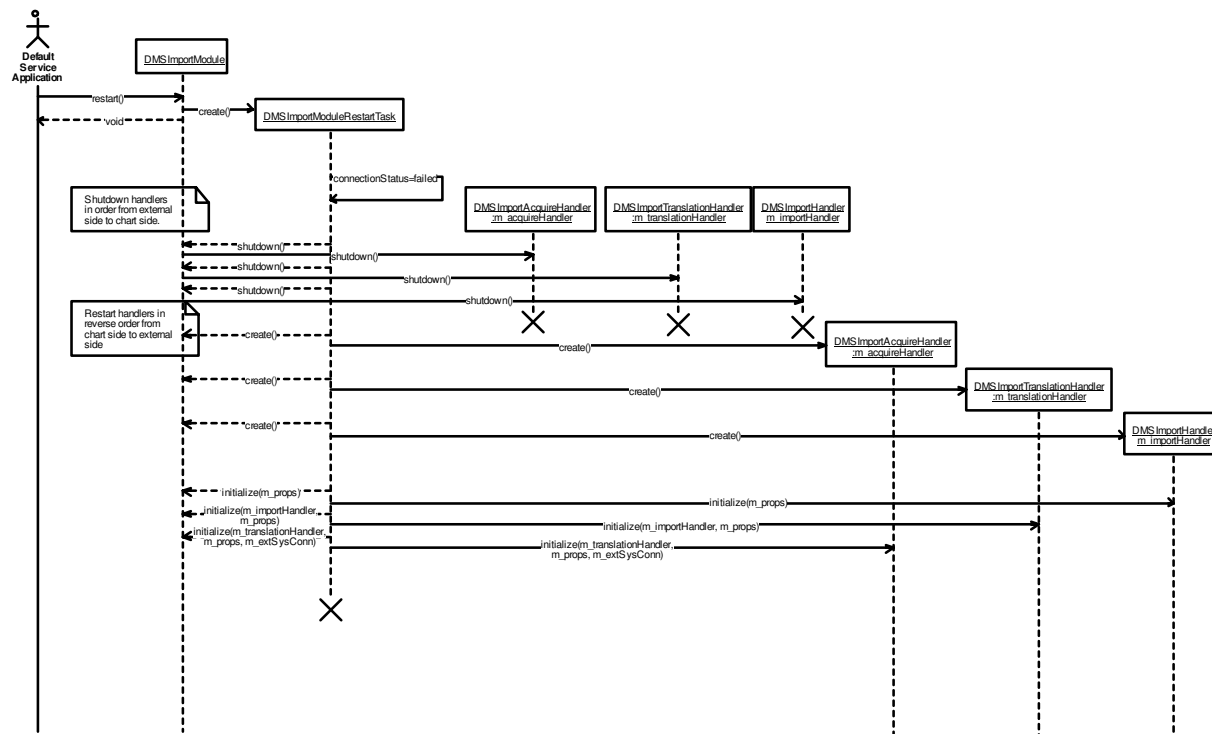


Figure 5-174 ExternalInterfaceModule:restartDMSImportModule (Sequence Diagram)

5.8.2.21 ExternalInterfaceModule:restartTSSImportModule (Sequence Diagram)

This diagram depicts the restart process of the TSSImportModule. It begins with the creation of a restart task that manages the restart process independent of the calling method. It is important to point out that the module itself does not restart as the method name might imply. Rather, the three event import data handlers, the TSSImportHandler, the TSSImportTranslationHandler, and the TSSImportAcquireHandler, are each shutdown and restarted with the intent of purging and resetting the data import queue. Order matters and the handlers are stopped in the reverse order from their original startup. During the time in which the data handlers are shutdown, the external connection status is forced to "failed" to bring about a connection reset with the external data source. Once this is complete, the restart task creates the event import data handlers anew. All three handlers being successfully created, they are, in turn, initialized so that they begin the external event import processing.

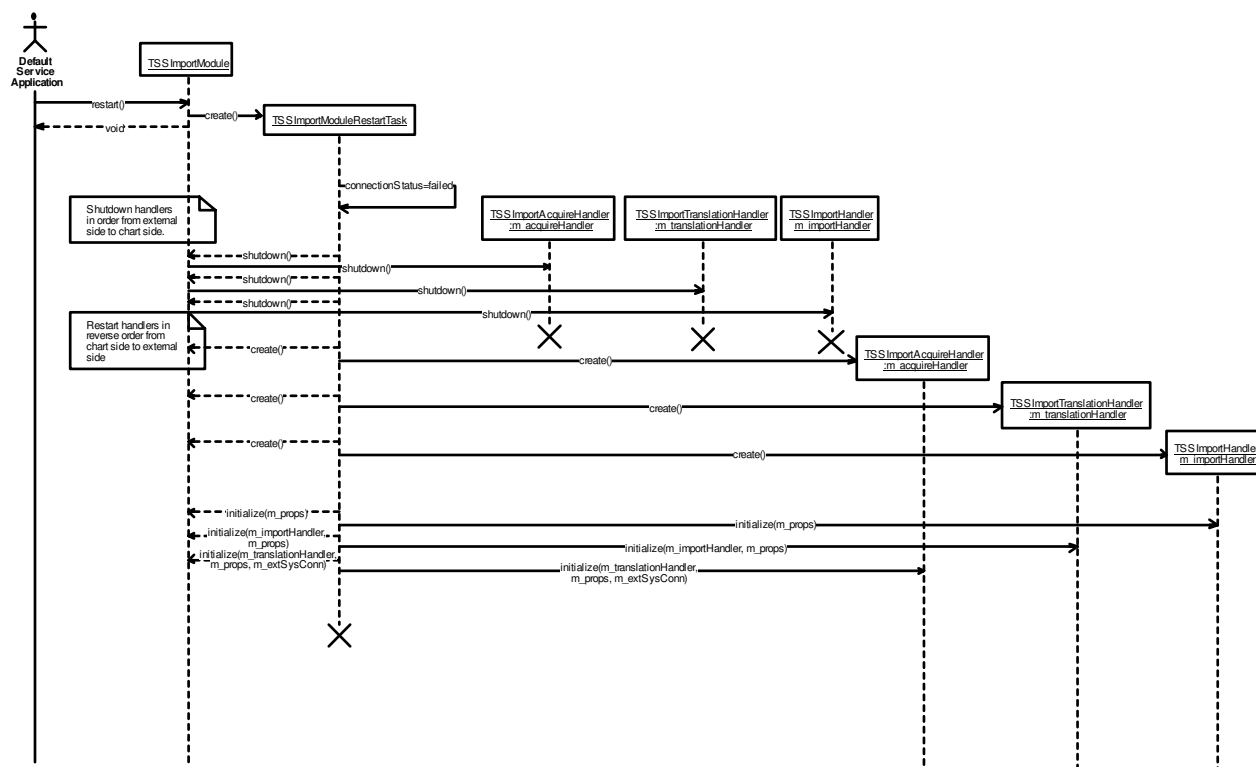


Figure 5-175 ExternalInterfaceModule:restartTSSImportModule (Sequence Diagram)

5.8.2.22 ExternalInterfaceModule:shutdownDMSImportModule (Sequence Diagram)

This diagram depicts the shutdown of the DMSImportModule. The startup thread is shutdown if for some reason it is still running. The ExtSysConnImpl is cleaned up accordingly. Note: in the shutdownTSSHandlers() method, handlers are shutdown in the reverse order they were originally initialized in.

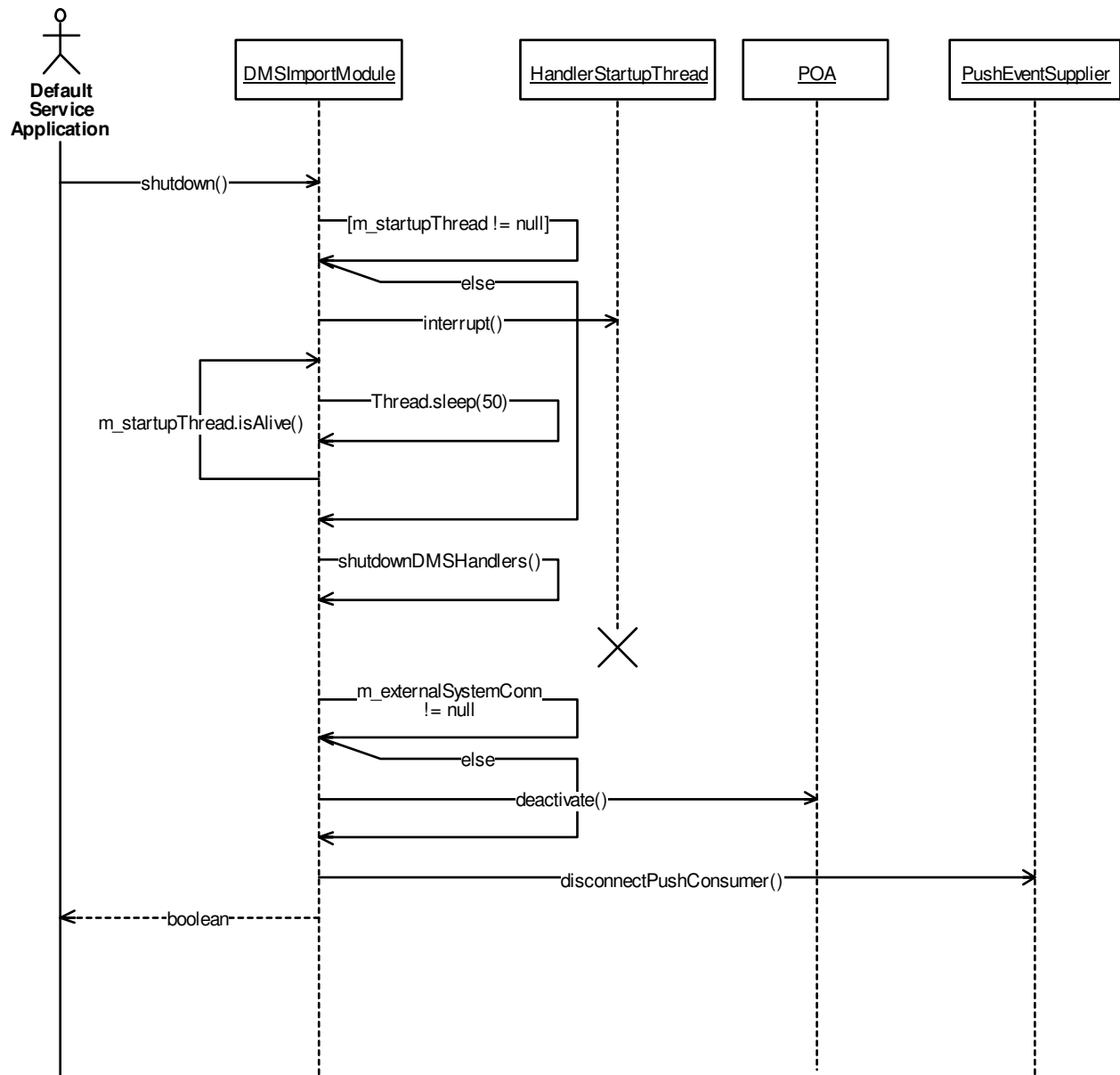


Figure 5-176 ExternalInterfaceModule:shutdownDMSImportModule (Sequence Diagram)

5.8.2.23 ExternalInterfaceModule:shutdownTSSImportModule (Sequence Diagram)

This diagram depicts the shutdown of the TSSImportModule. The startup thread is shutdown if for some reason it is still running. The ExtSysConnImpl is cleaned up accordingly. Note: in the shutdownTSSHandlers() method, handlers are shutdown in the reverse order they were originally initialized in.

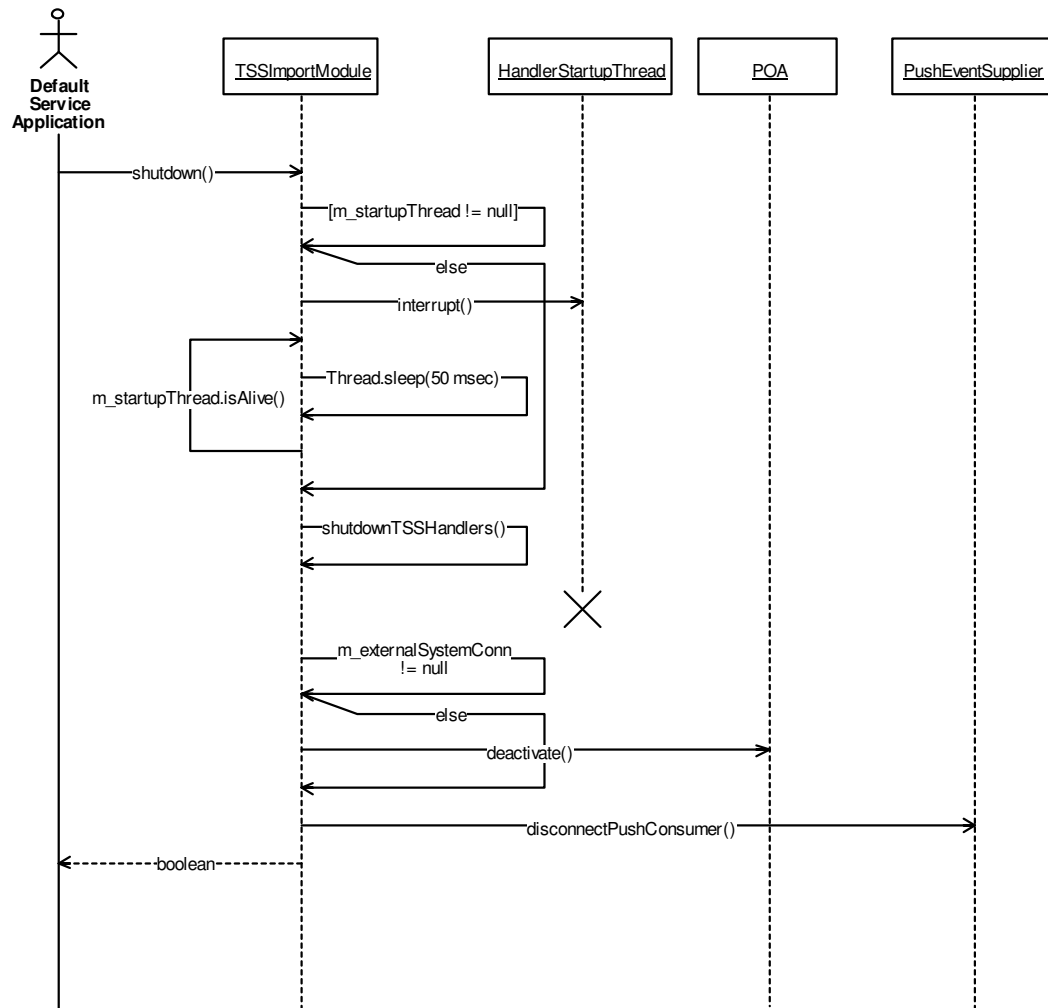


Figure 5-177 ExternalInterfaceModule:shutdownTSSImportModule (Sequence Diagram)

5.8.2.24 ExternalInterfaceModule:tssTranslationStep1Translate (Sequence Diagram)

This diagram depicts the translation of external TSS messages into Chart TSS data. The `TssTmddTranslationStep1.translate()` method is called by the `TSSImportTranslationHandler.handleTranslation()` method. It takes in a `AVList` and based on the message name processes it accordingly. For TSS related messages it pulls the XML string from the first element in the `AVList`. This xml string contains information for one specific TSS (inventory or status). It transforms the XML into a CHART-specific XML string using XSL Transformation. The resulting XML string is then added the outbound `AVList` and the method returns. For REFRESH messages no translation is done. The message is just added to the outbound `AVList` and the method returns.

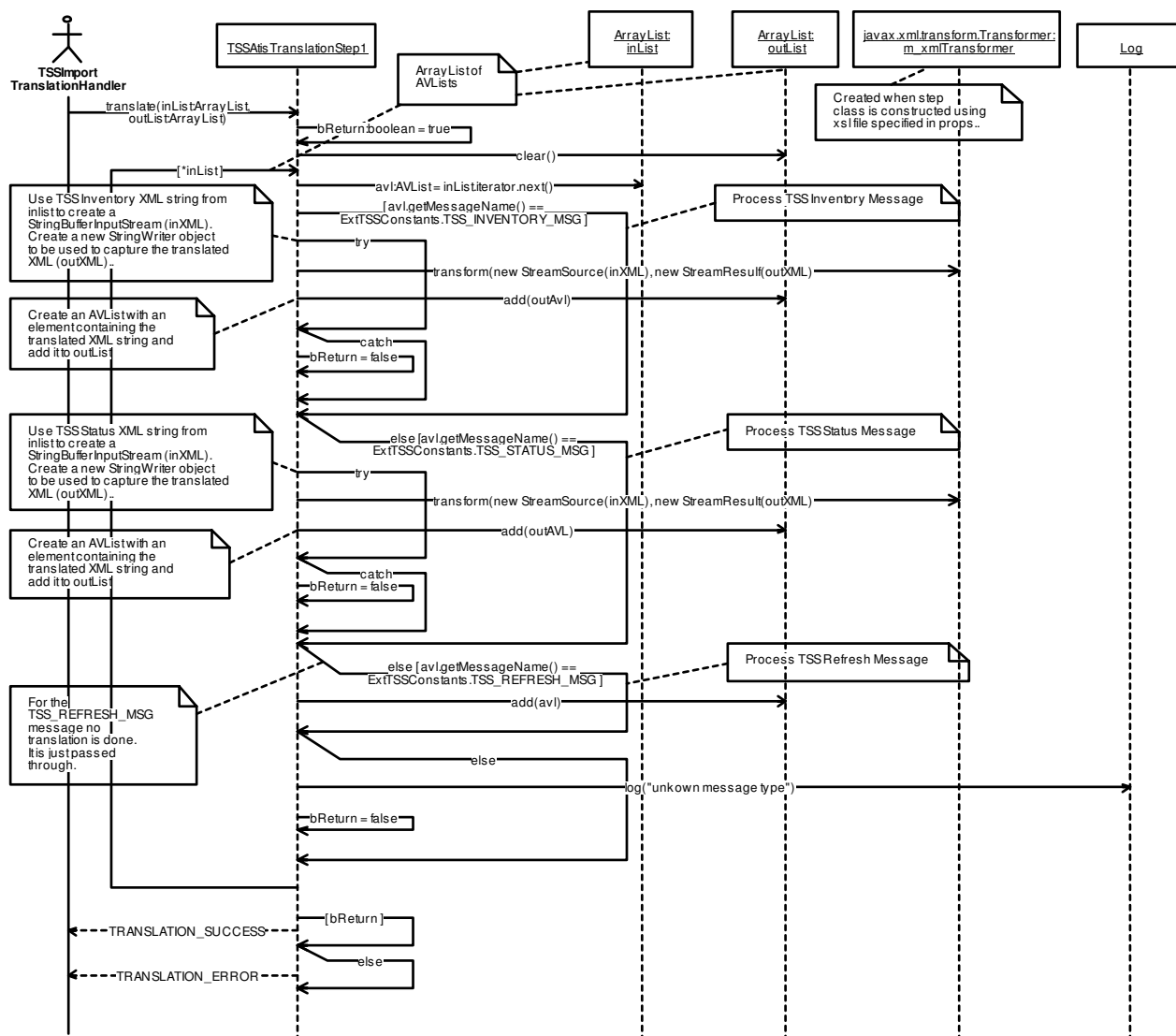


Figure 5-178 ExternalInterfaceModule:tssTranslationStep1Translate (Sequence Diagram)

5.8.2.25 ExternalSystemConnectionImpl:init (Sequence Diagram)

This diagram depicts the initialization of the ExternalSystemConnectionImpl. First, the AlertFactoryWrapper, NotificationManagerWrapper, and SystemProfileProperties private members populated along with the remain members which are passed in as arguments. A java.util.Timer object is created to support one or more TimerTasks. A TimerTask is created/scheduled to push out ExternalSystemConnection status events on a regular interval if configured to do so (configured statically in props file). A TimerTask is created/schedule to monitor external system connection alert/notification setting (System Profile Properties) and conditionally create notifications and ExternalSystemConnection alerts if connection goes in to a Failure or Warning state for a configurable amount of time.

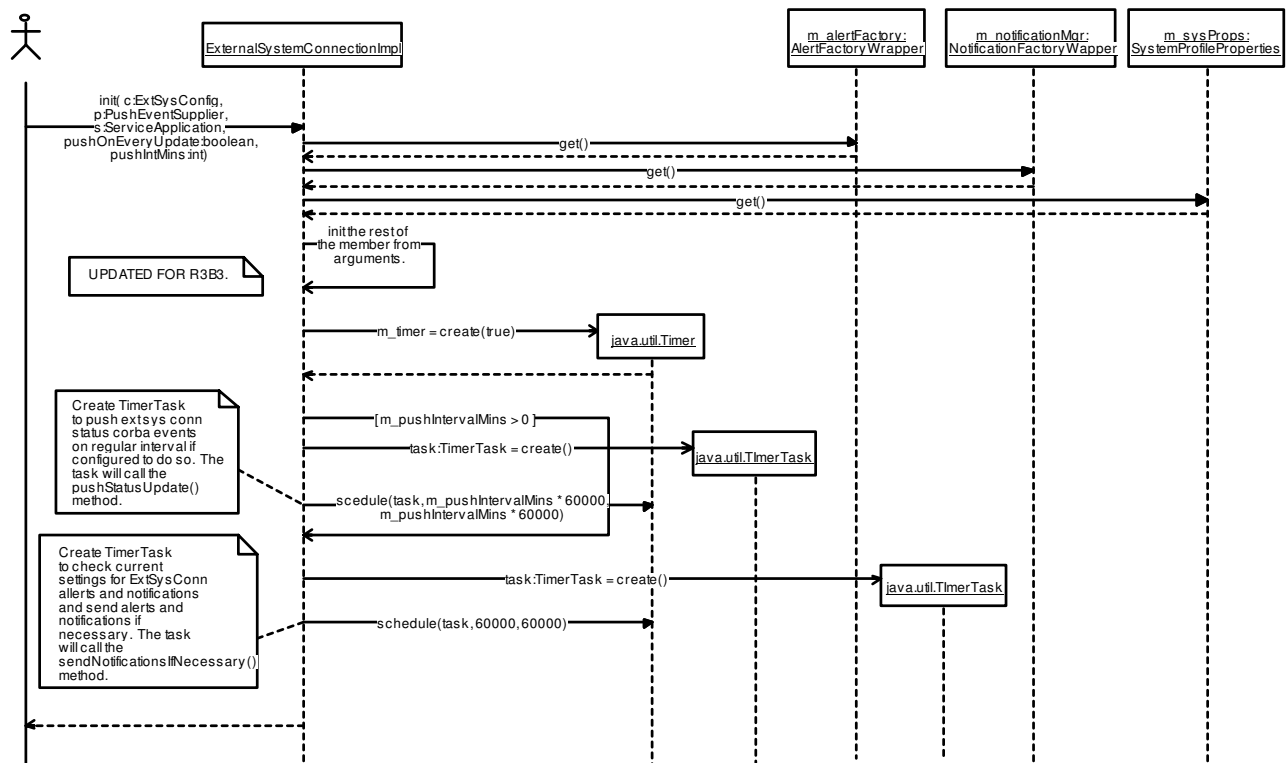


Figure 5-179 ExternalSystemConnectionImpl:init (Sequence Diagram)

5.8.2.26 ExternalSystemConnectionImpl:sendNotificationsIfNecessary (Sequence Diagram)

The ExternalSystemConnectionImpl.sendNotificationsIfNecessary() method is called from a TimerTask every minute to determine if notifications / alerts need to be created to convey Failure/Warning information about the connection represented by the impl.

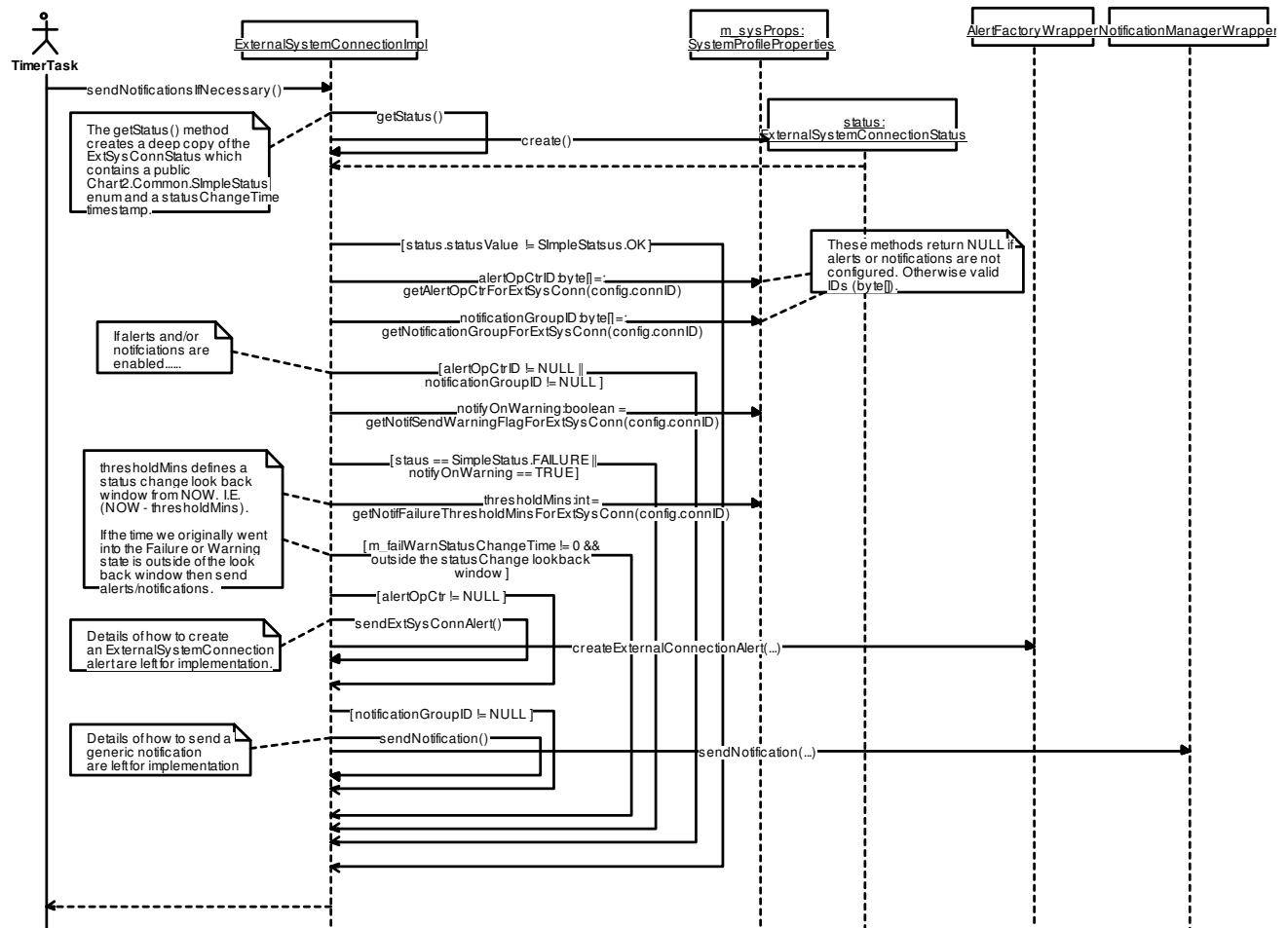


Figure 5-180 ExternalSystemConnectionImpl:sendNotificationsIfNecessary (Sequence Diagram)

5.8.2.27 TSSImportAcquireTask:execute (Sequence Diagram)

This diagram depicts the behavior of the execute() method of the EventImportAcquireTask. It is called when this task reaches the head of the Acquire command queue. This method is responsible for breaking the external message into individual traffic event messages. A special case it handles is that it must place a marker message in the message stream after the complete set of refresh events are received after a reconnect with RITIS. After the Translation Handler passes this marker to the Event Import handler, the Event Import handler knows that all traffic events have recently been refreshed and it should close all events whose last update time is older than the time in the marker message

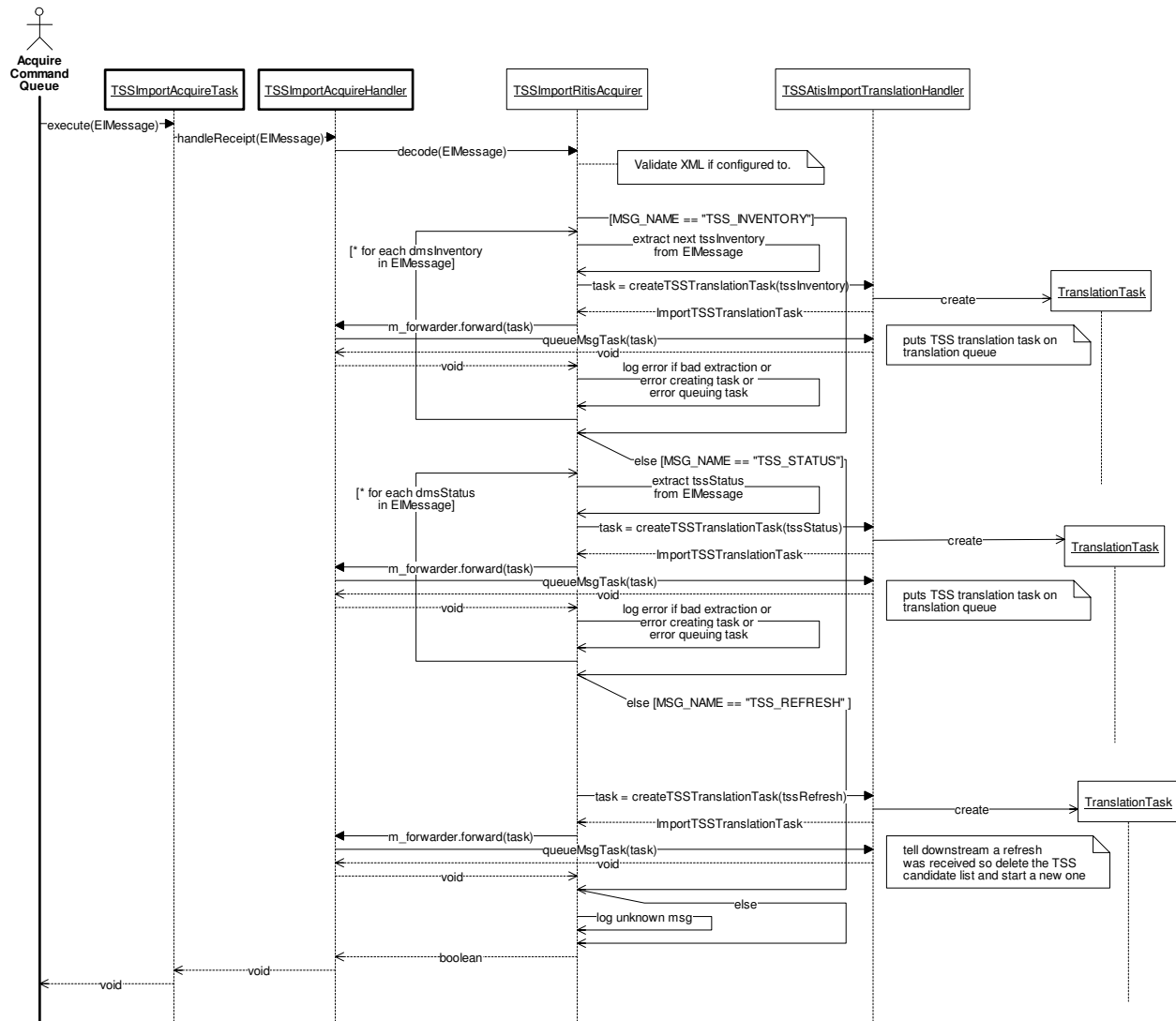


Figure 5-181 TSSImportAcquireTask:execute (Sequence Diagram)

5.8.2.28 TSSImportRitisAcquirer:connectIfNecessary (Sequence Diagram)

This diagram depicts the `connectIfNecessary()` method of the `TSSImportRitisAcquirer` class. This method is called at initiation and periodically after that. When invoked, if the RITIS connection is failed or stale (no activity on the connection for while - see properties), it attempts a reconnection. Its primary duty is to rebuild the JMS connection with RITIS by registering to have the `OnMessage()` method called whenever RITIS has a message to send. It is expected that the stale connection value will be large enough to not present a performance problem for either CHART or RITIS but small enough to be responsive to users if the connection is temporarily lost. The RITIS connection status is updated if a reconnect is attempted.

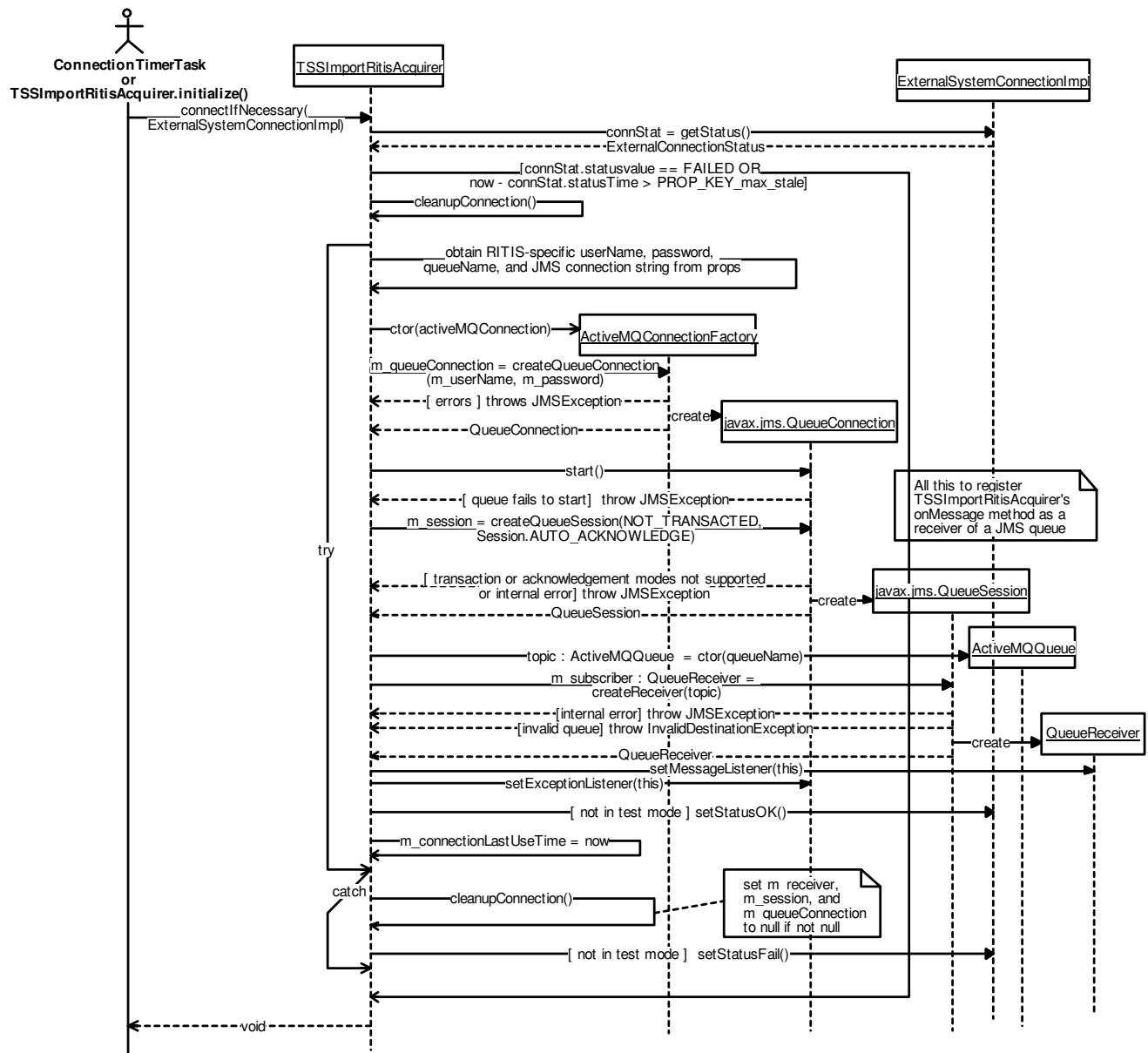


Figure 5-182 TSSImportRitisAcquirer:connectIfNecessary (Sequence Diagram)

5.8.2.29 TSSImportRitisAcquirer:initialize (Sequence Diagram)

This diagram depicts the initialize() method of the TSSImportRitisAcquirer class. After attempting to connect to RITIS and updating the connection status, it kicks off a periodic task that checks the connection status (see connectIfNecessary()) and reconnects, if necessary.

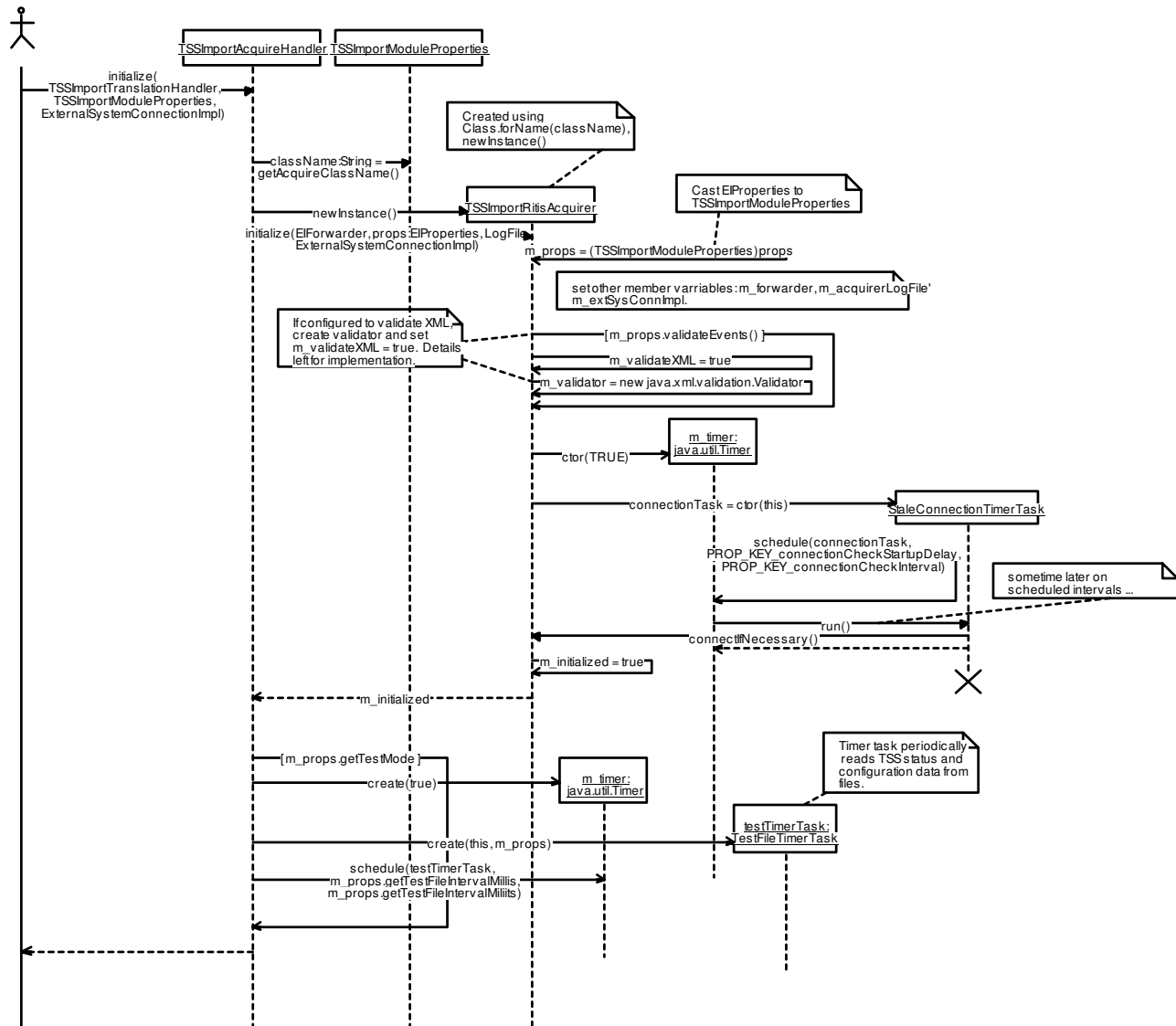


Figure 5-183 TSSImportRitisAcquirer:initialize (Sequence Diagram)

5.8.2.30 TSSImportRitisAcquirer:onMessage (Sequence Diagram)

This diagram depicts the onMessage() method of the TSSImportRitisAcquirer class. This method is called when RITIS has a TSS related message for CHART. To ensure CHART is responsive to RITIS, the onMessage() method's only job is to create an Acquire task containing the external message and put it on the Acquire command queue in the proper format and be ready for the next message from RITIS.

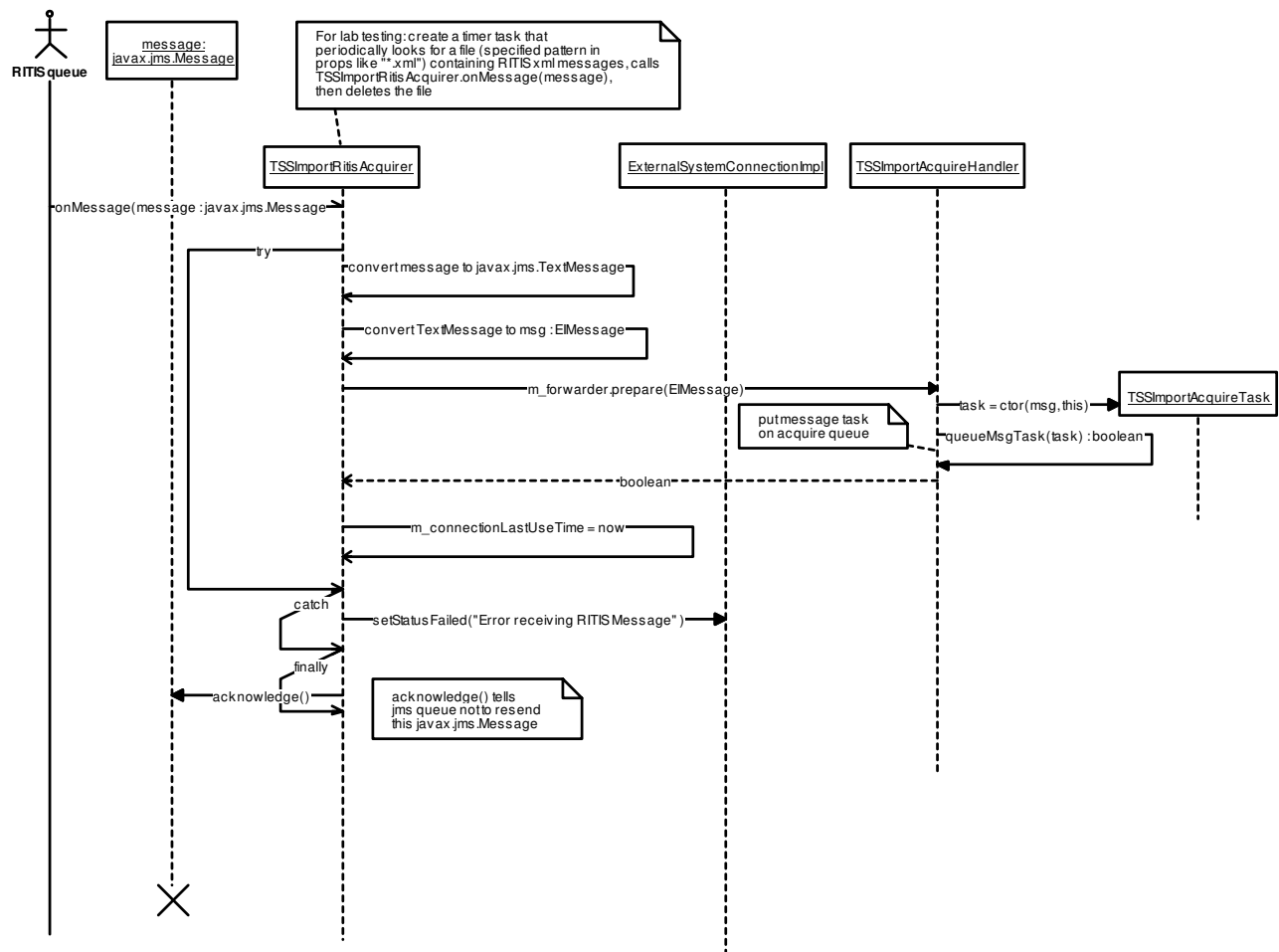


Figure 5-184 TSSImportRitisAcquirer:onMessage (Sequence Diagram)

5.9 GeoAreaModulePkg

5.9.1 Classes

5.9.1.1 GeoAreaModulePkg (Class Diagram)

This class diagram identifies the classes in the GeoAreaModule package. There are dependencies on CHART utility classes as well as code generated from CHART IDL

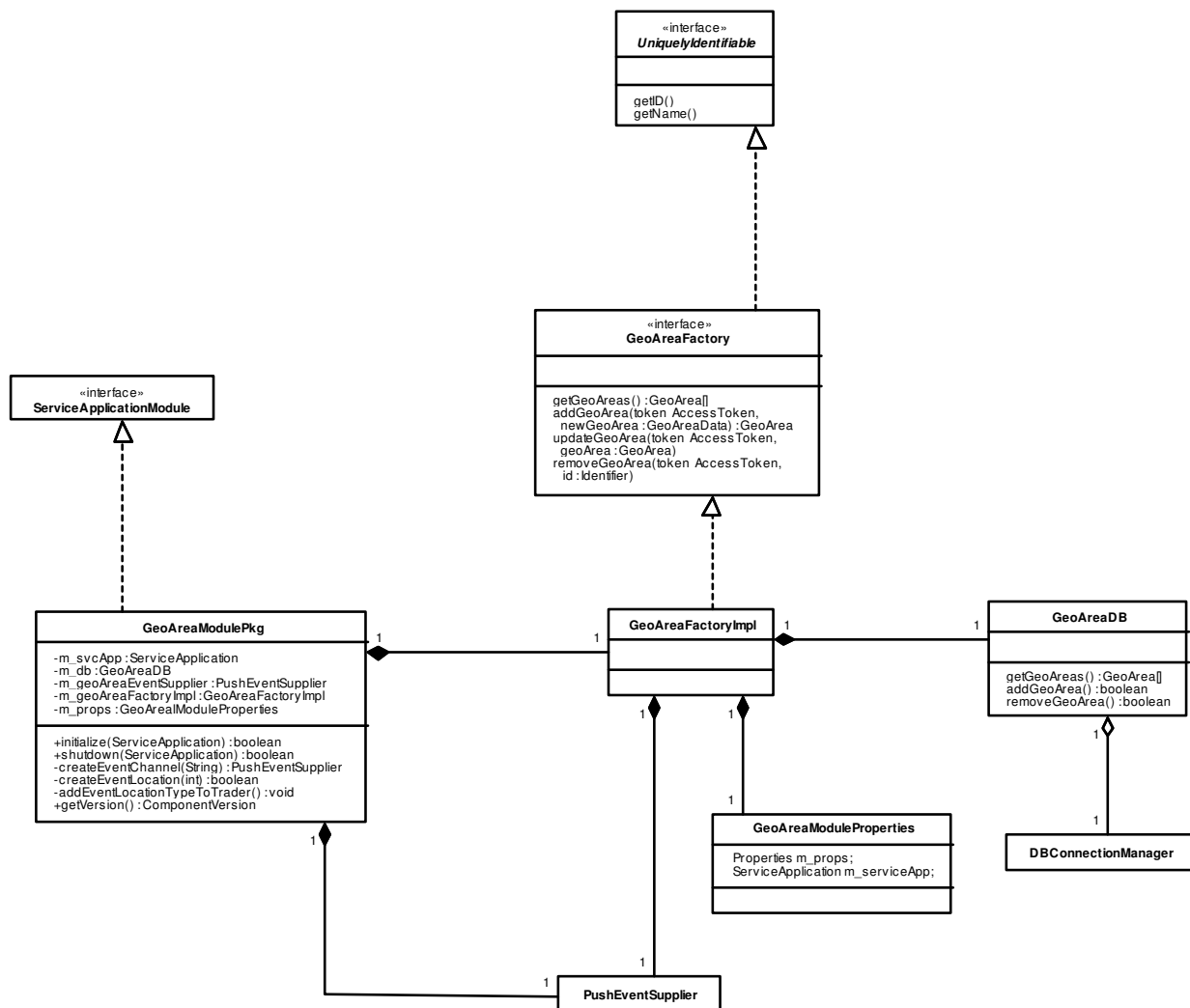


Figure 5-185 GeoAreaModulePkg (Class Diagram)

5.9.1.1.1 DBConnectionManager (Class)

This class implements a database connection manager that manages a pool of database connections. Any CHART II system thread requiring database access gets a database connection from the pool of connections maintained by this manager class. The connections are maintained in two separate lists namely, inUseList and freeList. The inUseList contains connections that have already been assigned to a thread. The freeList contains unassigned connections. This class assumes that an appropriate JDBC driver has been loaded either by using the "jdbc.drivers" system property or by loading it explicitly. The class has a monitor thread that is started by the constructor. This connection monitor thread periodically checks the inuseList to see if there are connections that are owned by dead threads and move such connections to the freeList. The connection monitor thread is started only if a non-zero value is specified for the monitoring time interval in the constructor.

5.9.1.1.2 GeoAreaDB (Class)

The GeoAreaDB class provides database access for creating, removing and updating GeoArea's in the Chart2 System.

5.9.1.1.3 GeoAreaFactory (Class)

This interface defines a factory responsible for managing GeoAreas with the CHART system.

5.9.1.1.4 GeoAreaFactoryImpl (Class)

This is the implementation of the GeoAreaFactory interface. It is responsible for handling requests related to GeoAreas within the CHART system.

5.9.1.1.5 GeoAreaModulePkg (Class)

This class provides the resources and support functionality necessary to serve GeoAreaFactory objects in a service application. It implements the ServiceApplicationModule interface which allows it to be installed as part of a DefaultServiceApplication.

5.9.1.1.6 GeoAreaModuleProperties (Class)

This class represents the configurable properties of the GeoAreaModule.

5.9.1.1.7 PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used.

This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.9.1.1.8 ServiceApplicationModule (Class)

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

5.9.1.1.9 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.9.2 Sequence Diagrams

5.9.2.1 GeoAreaFactoryImpl:addGeoArea (Sequence Diagram)

This sequence diagram depicts the IDL method addGeoArea(). If the user does not have functional rights to perform the action an AccessDenied exception is thrown. The GeoArea is then persisted and a GeoAreaAdded event is pushed. An Operations Log Record is added for this operation.

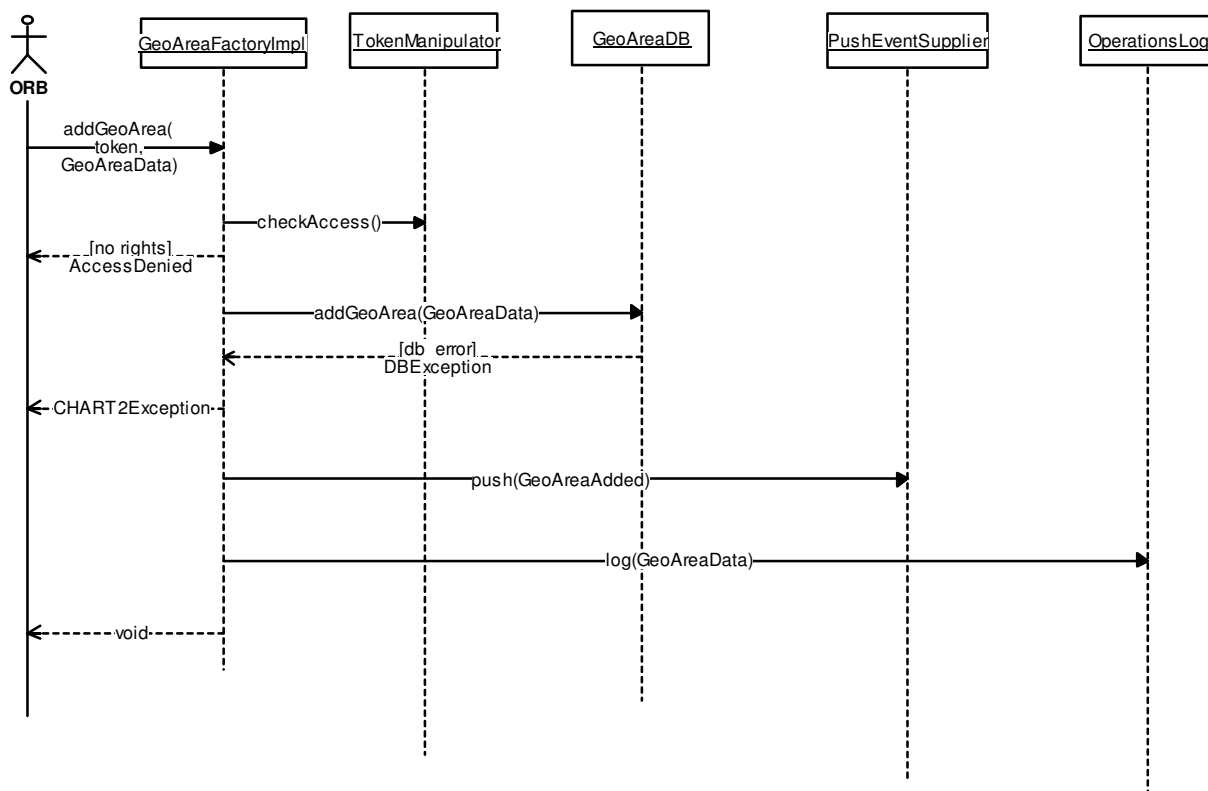


Figure 5-186 GeoAreaFactoryImpl:addGeoArea (Sequence Diagram)

5.9.2.2 GeoAreaFactoryImpl:getGeoAreas (Sequence Diagram)

This sequence diagram depicts the IDL method `getGeoAreas()`. If the user does not have functional rights to perform the action an `AccessDenied` exception is thrown. All of the `GeoArea`'s in the database are depersisted and returned to the ORB

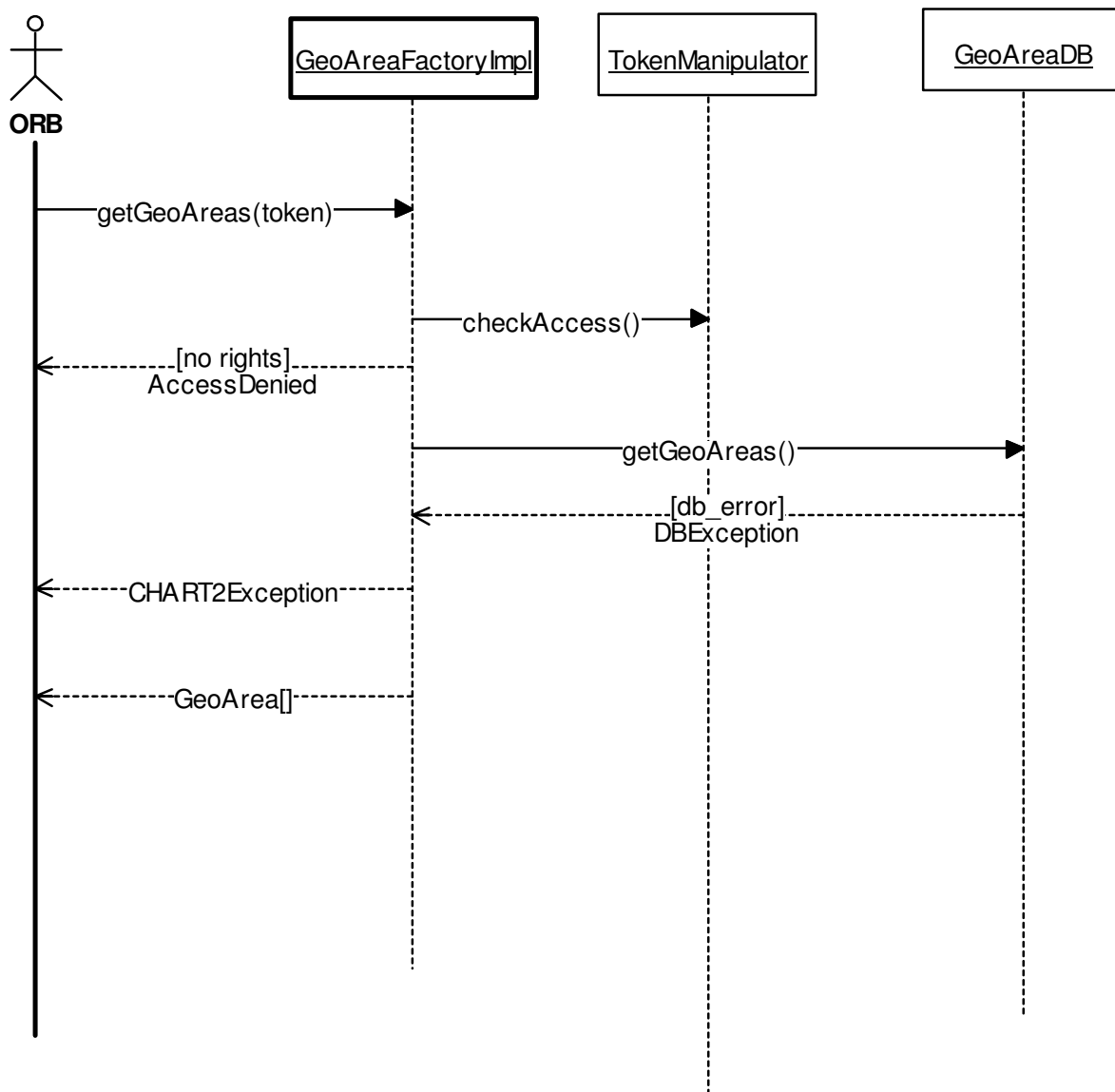


Figure 5-187 GeoAreaFactoryImpl:getGeoAreas (Sequence Diagram)

5.9.2.3 GeoAreaFactoryImpl:removeGeoArea (Sequence Diagram)

This sequence diagram depicts the IDL method removeGeoArea(). If the user does not have functional rights to perform the action an AccessDenied exception is thrown. The GeoArea is then deleted from the database and a GeoAreaRemoved event is pushed. An Operations Log Record is added for this operation.

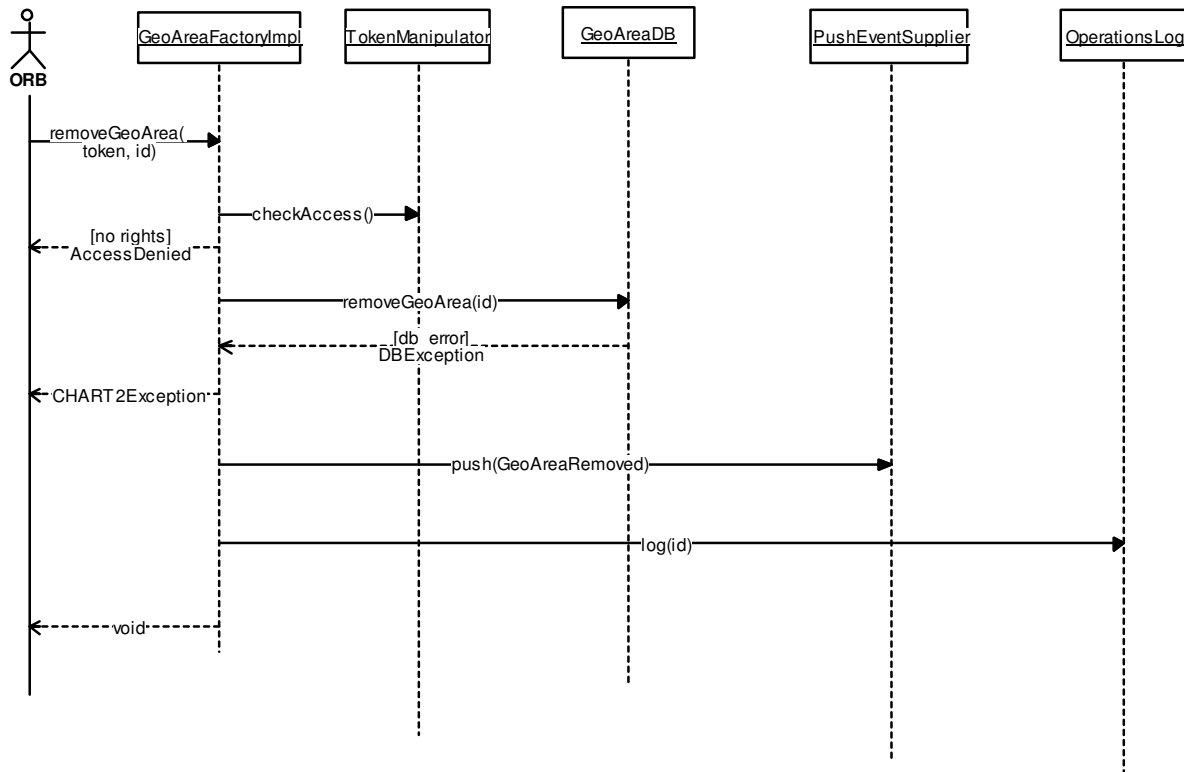


Figure 5-188 GeoAreaFactoryImpl:removeGeoArea (Sequence Diagram)

5.9.2.4 GeoAreaFactoryImpl:updateGeoArea (Sequence Diagram)

This sequence diagram depicts the IDL method `updateGeoArea()`. If the user does not have functional rights to perform the action an `AccessDenied` exception is thrown. The `GeoArea` is then persisted and a `GeoAreaUpdated` event is pushed. An Operations Log Record is added for this operation.

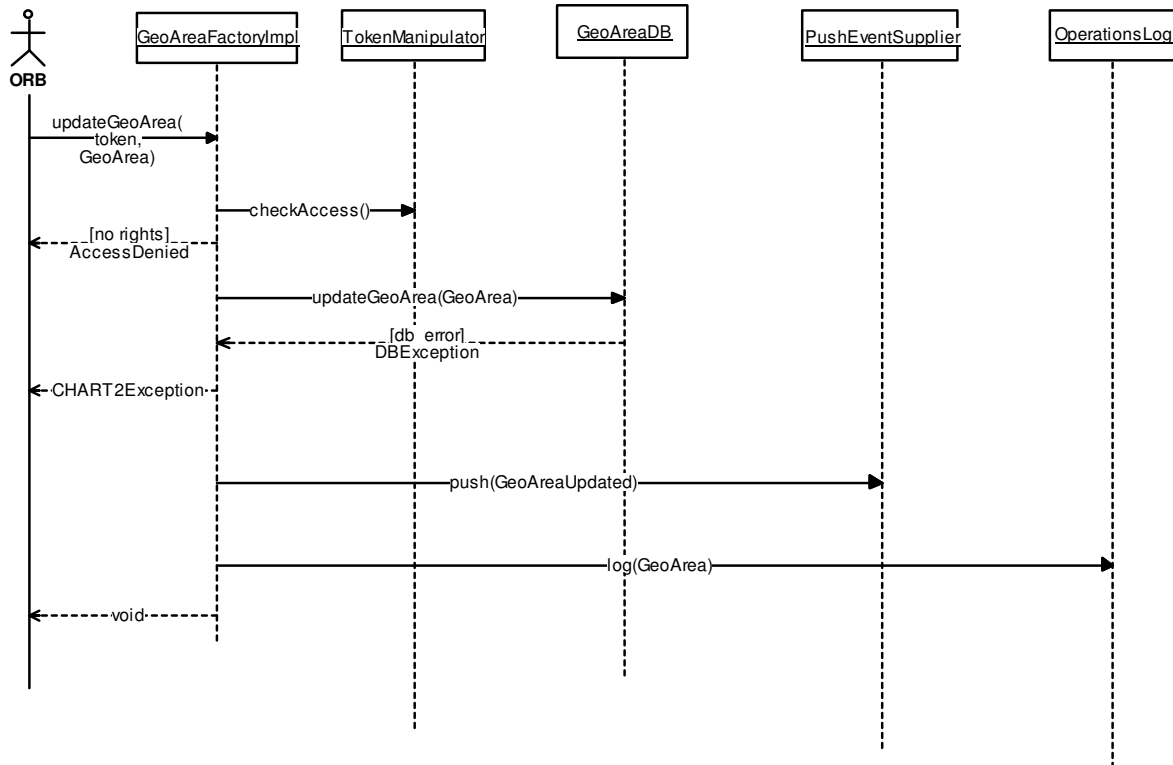


Figure 5-189 GeoAreaFactoryImpl:updateGeoArea (Sequence Diagram)

5.9.2.5 GeoAreaModulePkg:Initialize (Sequence Diagram)

This diagram shows what happens when the GeoAreaModule is initialized. The ServiceApplication calls the GeoAreaModule to initialize, which reads in the properties from a file, overriding the default properties. It creates an event channel for GeoAreas and publishes the channel in the trading service so that other applications can see it. It creates a GeoAreaDB object to handle all of the database calls, and a GeoAreaFactoryImpl object to manage the GeoAreas. The GeoAreaFactory is exported to the trading service and the GeoAreaModule initialize method returns.

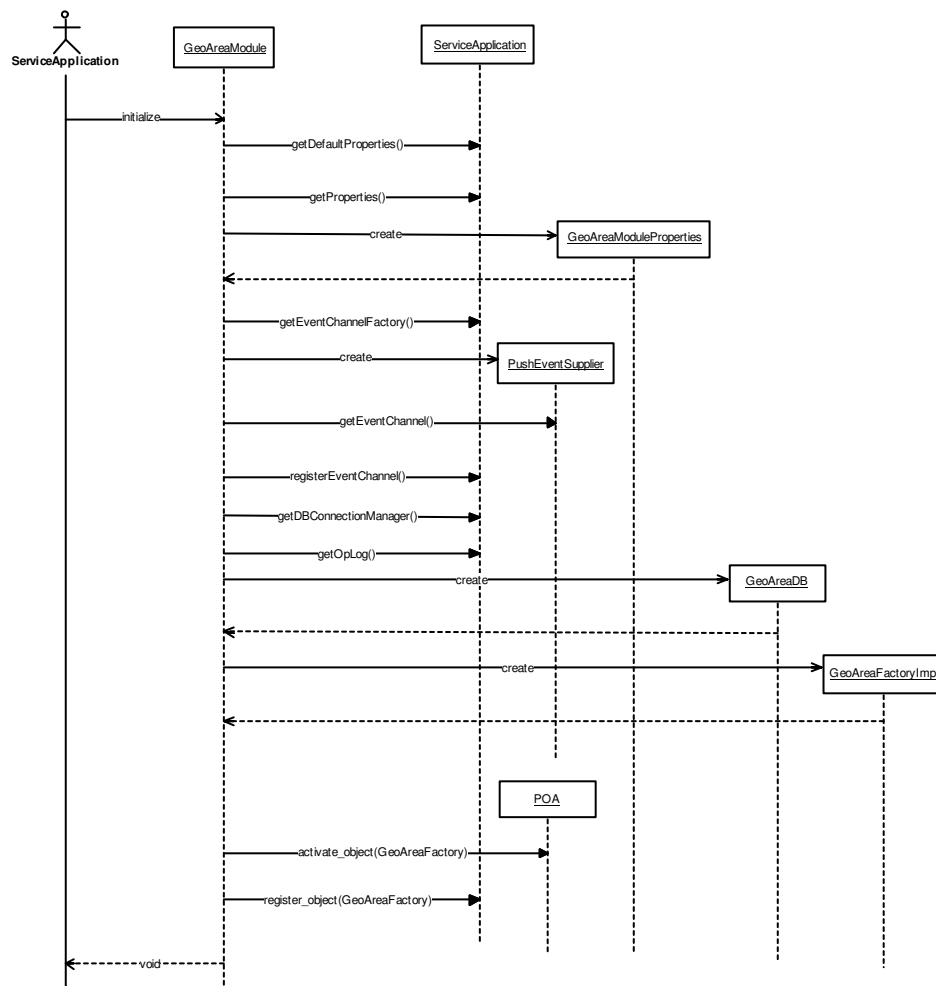


Figure 5-190 GeoAreaModulePkg:Initialize (Sequence Diagram)

5.9.2.6 GeoAreaModulePkg:Shutdown (Sequence Diagram)

When the GeoAreaModule is shut down by the ServiceApplication, it disconnects its objects from the ORB, and releases any resources it is using.

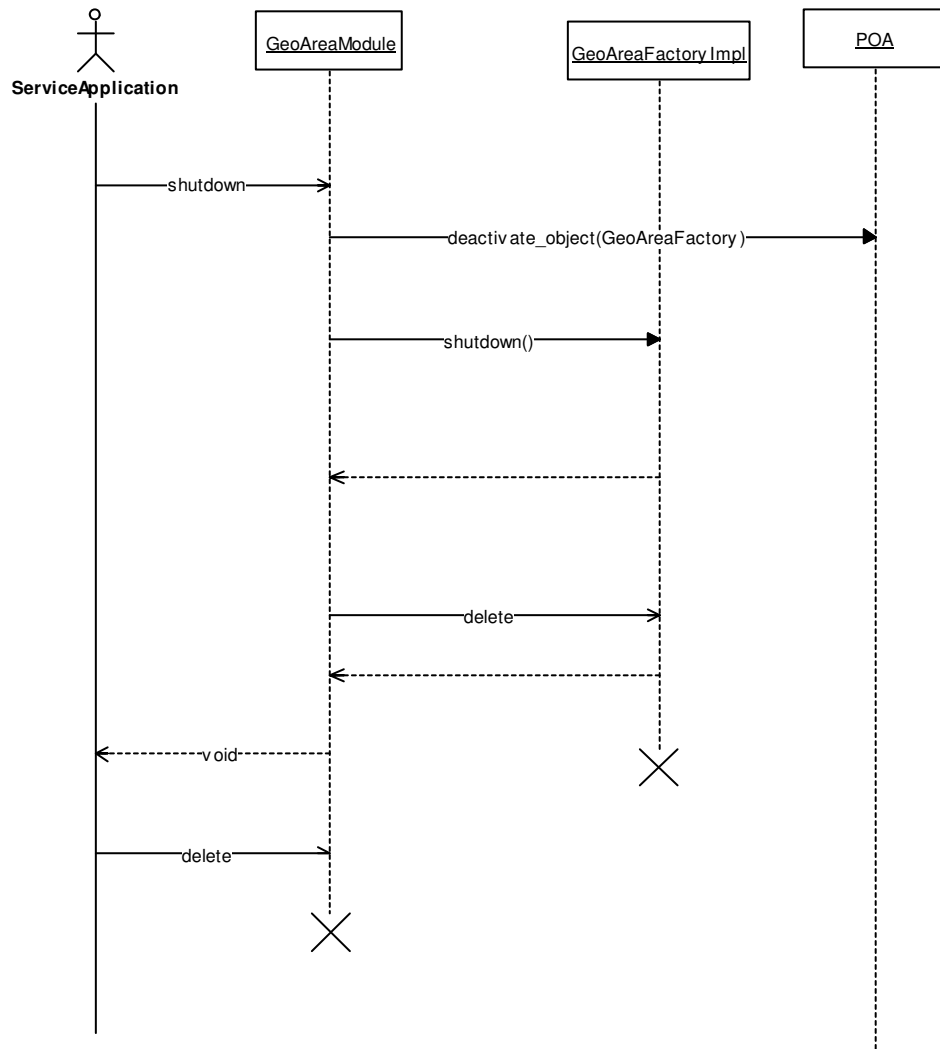


Figure 5-191 GeoAreaModulePkg:Shutdown (Sequence Diagram)

5.10 HARControlModulePkg

5.10.1 Classes

5.10.1.1 HARControlModule (Class Diagram)

This class diagram shows classes that support the use of Highway Advisory Radio (HAR) devices in the Chart II system. Details are only shown for classes that exist specifically for HAR control. Auxiliary classes used from other various utility or system interface packages are shown by name only.

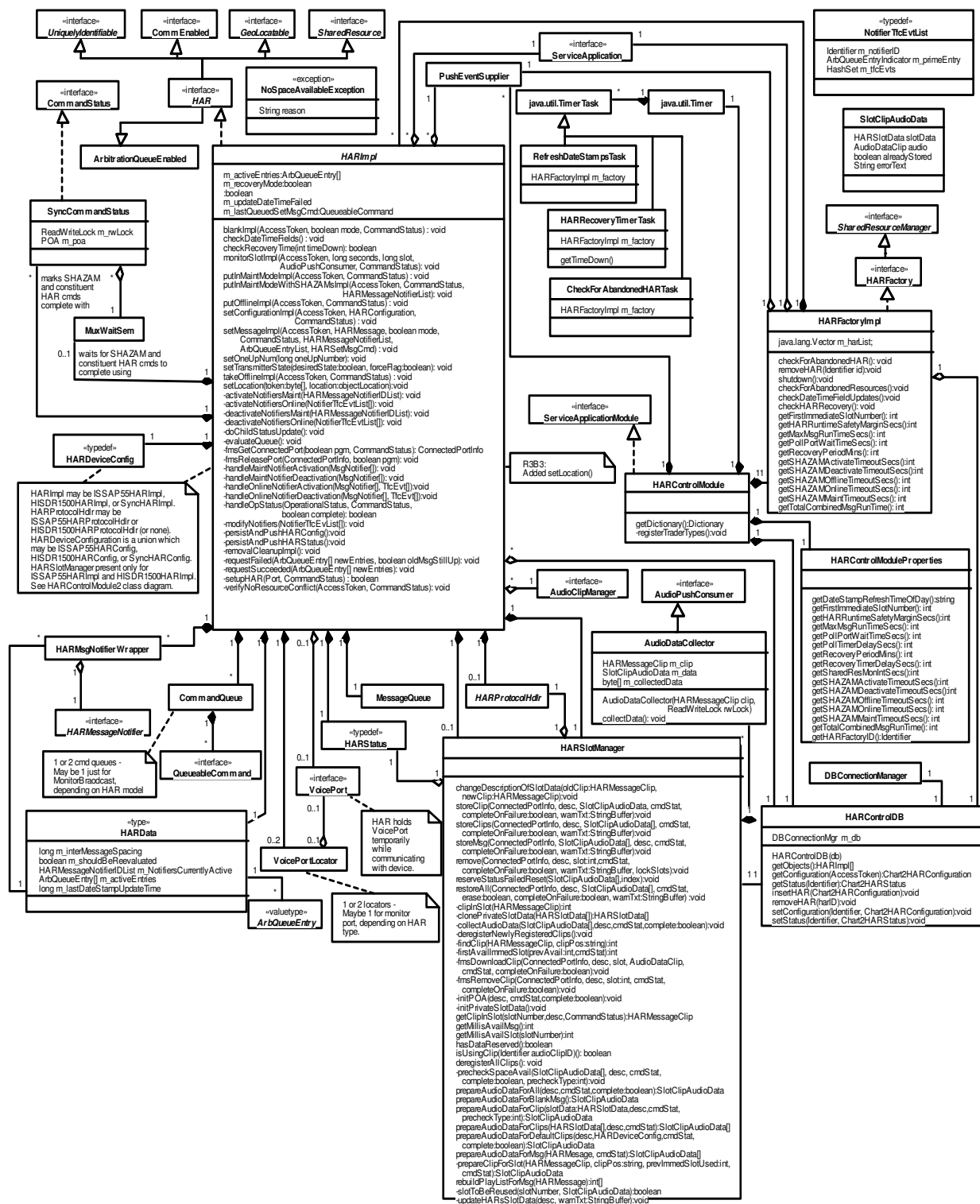


Figure 5-192 HARControlModule (Class Diagram)

5.10.1.1.1 ArbitrationQueueEnabled (Class)

ArbQueueEnabled is a synonym for ArbitrationQueue. An ArbitrationQueue is a queue that arbitrates the usage of a device. The evaluation of the queue determines which message(s) should be on the device, based upon the priority of the queue entries. When entries are added to the queue, they are assigned a priority level based on the type of traffic event with which they are associated, and also upon the current contents of the queue. The priority of the queue entries can be modified after they are added to the queue. The queue is evaluated when the device is online and queue entries are added or removed, when an entry's priority is modified, or when the device is put online.

5.10.1.1.2 ArbQueueEntry (Class)

This class is used for an entry on the arbitration queue, for a single message, and for a single traffic event. (It is possible, in the case of HARNotifierArbQueueEntry objects, that certain ArbQueueEntries can be on behalf of multiple TrafficEvents. In such cases, one TrafficEvent among all those involved is picked to be the responsible TrafficEvent stored in m_indicator, the ArbQueueEntryIndicator for the entry.)

5.10.1.1.3 AudioClipManager (Class)

This interface provides a way to store audio data associated with HARMessageAudioDataClip objects, converting the HARMessageAudioDataClip objects to HARMessageAudioClip objects in the process. The HARMessageAudioClip objects are created with a reference back to the AudioClipManager in them, so that the audio clips themselves can provide access to the audio data (through their stream() interface), by contacting the AudioClipManager (an AudioClipStreamer) to stream the data. The AudioClipManager also provides a capability for various AudioClipOwners to register and deregister their "interest" in a specific clip. When a clip no longer has any interested owners, it can be (and is) deleted from the database.

5.10.1.1.4 AudioDataCollector (Class)

This object is used to stream a HARMessageClip and write the streamed audio .wav data to a .wav file. It is used as a utility by the HARSlotManager to prepare HARMessageClips for download into the HAR (which is accomplished via the ISSAP55HARProtocolHdlr by passing the file name of the .wav file into it).

5.10.1.1.5 AudioPushConsumer (Class)

This interface is implemented by objects that wish to receive audio data using the push model, where the server pushes the data to the consumer. One call to pushAudioProperties() will always precede any calls to pushAudio(). When the AudioClipStreamer is done sending data in pushAudio() calls, it sends a pushCompleted() to indicate successful completion, or a pushFailure() to indicate a failure which has prevented the streaming from completing. PushAudio() returns a boolean "continue" flag, which, if returned as false, indicates that the consumer no longer wants to continue receiving audio data. In this case, the stream stops pushing data immediately, with no call

to pushCompleted() or pushFailure() necessary.

5.10.1.1.6 CheckForAbandonedHARTask (Class)

This class is a timer task that is executed periodically by a timer. When the run method in this class is called, it calls the HARFactoryImpl's checkForAbandonedResources() method, which causes the factory to evaluate each HAR in the factory and issue an abandoned resource event for any HARs which have a controlling op center with no users logged in.

5.10.1.1.7 CommandQueue (Class)

The CommandQueue class provides a queue for QueueableCommand objects. The CommandQueue has a thread that it uses to process each QueueableCommand in a first in first out order. As each command object is pulled off the queue by the CommandQueue's thread, the command object's execute method is called, at which time the command performs its intended task.

5.10.1.1.8 CommandStatus (Class)

The CommandStatus CORBA interface is used to allow a calling process to be notified of the progress of a long-running asynchronous operation. This is normally used when field communications are involved to complete a method call. The most common use is to allow a GUI to show the user the progress of an operation. It can also be used and watched by a server process when it needs to call on another server process to complete an operation. The long running operation typically calls back to the CommandStatus object periodically as the command is being executed, to provide in-progress status information, and it always makes a final call to the CommandStatus when the operation has completed. The final call to the CommandStatus from the long running operation indicates the success or failure of the command.

5.10.1.1.9 CommEnabled (Class)

The CommEnabled interface is implemented by objects that can be taken offline, put online, or put in maintenance mode through a standard interface. These states typically apply only to field devices. When a device is taken offline, it is no longer available for use through the system and automated polling (if any) is halted. When put online, a device is again available for use by TrafficEvents within the system and automated polling is enabled (if applicable). When put in maintenance mode a device is offline (i.e., cannot be used by TrafficEvents), and maintenance commands appropriate for the particular type of device are allowed to help in troubleshooting.

5.10.1.1.10 DBConnectionManager (Class)

This class implements a database connection manager that manages a pool of database connections. Any CHART II system thread requiring database access gets a database connection from the pool of connections maintained by this manager class. The connections are maintained in two separate lists namely, inUseList and freeList. The inUseList contains connections that have already been assigned to a thread. The freeList contains unassigned

connections. This class assumes that an appropriate JDBC driver has been loaded either by using the "jdbc.drivers" system property or by loading it explicitly. The class has a monitor thread that is started by the constructor. This connection monitor thread periodically checks the inuseList to see if there are connections that are owned by dead threads and move such connections to the freeList. The connection monitor thread is started only if a non-zero value is specified for the monitoring time interval in the constructor.

5.10.1.1.11 GeoLocatable (Class)

This interface is implemented by objects that can provide location information to their users.

5.10.1.1.12 HAR (Class)

This class is used to represent a Highway Advisory Radio (HAR) device. A HAR is used to broadcast traffic related information over a localized radio transmitter, making the information available to the traveler. This interface contains methods for getting and setting configuration, getting status, changing communications modes of a HAR, and manipulating and monitoring the HAR in maintenance and online modes.

5.10.1.1.13 HARControlDB (Class)

This class contains all the database interaction for the HARControlModule. This class provides the ability to retrieve all HAR information on initialization, update of the configuration and status information, and insert or remove a HAR device from the system.

5.10.1.1.14 HARControlModule (Class)

This class implements the ServiceApplicationModule interface, providing a platform for publishing HAR objects and the HARFactory object within a service application. This class is the controlling class for the HAR module, providing for the initialization and overall operation of the module. This class creates and starts the timer tasks necessary for refreshing timestamps on the HAR, checking for abandoned shared resources, and recovery processing.

5.10.1.1.15 HARControlModuleProperties (Class)

This class contains settings from a properties file used to specify parameters to be used by objects within the HARControlModule for the current instance of the application. These settings are read during the module initialization. The module must be restarted to apply any changes made to the properties file.

5.10.1.1.16 HARData (Class)

This class is used to store and persist data pertaining to a HAR which is not part of the HARStatus (i.e., not transmitted to clients in status updates or at any other time).

5.10.1.1.17HARDeviceConfig (Class)

HARDeviceConfig is a union which can contain the configuration for a ISS AP55 HAR, a HIS DR1500 HAR, or a Synchronizable HAR (a "virtual" HAR representing a collection of synchronized HARs). In R2B3 only DR1500 HARs are synchronizable.

5.10.1.1.18HARFactory (Class)

This CORBA interface allows new HAR objects to be added to the system. It also allows a requester to acquire a list of HAR objects under the domain of the specific HARFactory object.

5.10.1.1.19HARFactoryImpl (Class)

This class implements the HARFactory interface as defined by the IDL specified in the System Interfaces section. This class maintains the HAR objects served by this HAR service.

5.10.1.1.20HARImpl (Class)

This class implements HAR as defined by IDL specified in the System Interfaces section. Since there is only one model of HAR currently envisioned for CHART II, this HARImpl class is implementing the ISS AP55 HAR specifically.

5.10.1.1.21HARMessageNotifier (Class)

The HARMessageNotifier class specifies an interface to be implemented by devices that can be used to notify the traveler to tune in to a radio station to hear a traffic message being broadcast by a HAR. A HARMessageNotifier is directional and allows users of the device to better determine if activation of the device is warranted for the message being broadcast by the HAR. This interface can be implemented by SHAZAM devices and by DMS devices which are allowed to provide a SHAZAM-like message.

5.10.1.1.22HARMsgNotifierWrapper (Class)

This wrapper class is used to wrap HAR message notifiers associated with a HAR. This class handles finding the reference of the notifier object given only the object's ID. The object discovery is done at the point of first use or if a currently held reference produces a CORBA failure when used.

5.10.1.1.23HARProtocolHdlr (Class)

The HARProtocolHdlr implements is the base class to implement the commands which will actually be sent to the HAR..

5.10.1.1.24HARRecoveryTimerTask (Class)

This Timer Task runs on a regular basis (on the order of every 15-30 seconds) during the life of the process. During normal operations, this task's sole purpose is to write a

timestamp to a file each time it is called. This timestamp file serves to provide, to an approximation as accurate as its frequency of invocation, when the HARService last went down, an essential piece of information for recovery during HARService startup. When the HARService has recently started up, this Task, in addition to maintaining an up-to-date timestamp in the timestamp file, also calls a method in the Factory (checkHARRecovery) which requests all HAR objects to check and see if their recovery period has expired. (The recovery period is a system-wide constant, on the order of 10-15 minutes.) Each HAR terminates its recovery period as soon as all its TrafficEvents are resolved, or when the message queue is modified through an addEntry or changePriority call, or, if neither of those cases happens, at the end of the recovery period timer. (When all HARs have terminated their recovery period, checkHARRecovery is no longer called.)

When each HAR checks its own recovery time, if it finds that it has just now exceeded the recovery period, it calls its MessageQueue to take one last try at resolving traffic events on its queue, then the HAR makes final a determination as to what message (or blank) belongs on the sign, and it requests the HAR to set its message appropriately (either to the message(s) at the top of the queue, or to the default message, if no messages are queued.

5.10.1.1.25HARSlotManager (Class)

This class manages the slot usage for the HARImpl. When a clip is to be stored in the HAR controller, this class is called instead of calling the ISSAP55HARProtocolHdlr directly. This class ensures the reserved slot numbers (default header, default trailer, default message, immediate message slots) are not overlaid with other clips stored in the controller. When clips are stored in slots in the controller, this class keeps track of the run time for each and the total run time for the device and provides an error when the storage of a clip exceeds the configured available run time of the device.

This class also manages the condition when multiple slots are needed for the current (immediate) message. This will be true any time multiple messages are combined into one message on the HAR (up to the maximum play time for a combined message). A HAR has many immediate slots available for cases such as this.

5.10.1.1.26HARStatus (Class)

This class (struct) contains data that indicates the current status of a HAR device. The data contained in this class is that status information which can be transmitted from the HAR to the client as necessary. This struct is also used to within the HAR Service to transmit data to/from the HARControlDB database interface class. (The HAR implementation also contains other private status data elements which are not elements of this class.)

5.10.1.1.27java.util.Timer (Class)

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

5.10.1.1.28 java.util.TimerTask (Class)

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

5.10.1.1.29 MessageQueue (Class)

This class represents a message queue object. It will provide the ability to add, remove, and reprioritize traffic event entries in a prioritized list.

5.10.1.1.30 MuxWaitSem (Class)

This object is used block execution of a thread while it is running multiple long running commands which need to be waited on. This class watches the SyncCommandStatus of each command and releases control back to the main thread when all "child" long-running processes have completed their respective CommandStatus object.

5.10.1.1.31 NoSpaceAvailableException (Class)

This exception is thrown by the HARSlotManager when there is not enough room in the HAR to store the desired message as requested. This exception is local to the HAR service only. If the exception needs to propagate out to a user (GUI), it is converted to a CHART2Exception first. The distinction is required within the HAR service since a NoSpaceAvailableException is not to be considered a failure of the device or the communications.

5.10.1.1.32 NotifierTfcEvtList (Class)

This class is used to keep track of the relationships between HAR notifiers, and the traffic events which are requesting that they be activated. One traffic event is chosen to be the primary one, and is used as part of the ArbQueueEntryIndicator stored within this class. The m_primeEntry and m_tfcEvents are used as parameters to activate and/or modify the HAR notice on the notifier.

5.10.1.1.33 PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.10.1.1.34QueueableCommand (Class)

A QueueableCommand is an interface used to represent a command that can be placed on a CommandQueue for asynchronous execution. Derived classes implement the execute method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a CommandQueue. The CommandQueue driver calls the execute method to execute a command in the queue and a call to the interrupted method is made when a CommandQueue is shut down.

5.10.1.1.35RefreshDateStampsTask (Class)

This class is a timer task that is executed periodically by a timer. When executed, the run method of this class calls the HARFactoryImpl's checkDateTimeFieldUpdates(), which in turn calls each HAR in the factory to have it determine if it needs to update any field messages that use datestamp fields. These messages are reconverted to voice, and the datestamp tag, in the format "<DATESTAMP>" is replaced by text words for the day of week, month, and day of month (e.g. "Wednesday, July 14"). The reconverted messages are then queued to be resent to the HAR.

5.10.1.1.36ServiceApplication (Class)

This interface is implemented by objects that can provide the basic services needed by a ChartII service application. These services include providing access to basic CORBA objects that are needed by service applications, such as the ORB, POA, Trader, and Event Service.

5.10.1.1.37ServiceApplicationModule (Class)

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

5.10.1.1.38SharedResource (Class)

The SharedResource interface is implemented by any object that may have an operations center responsible for the disposition of the resource while the resource is in use.

5.10.1.1.39SharedResourceManager (Class)

The SharedResourceManager interface is implemented by classes that manage shared resources. Implementing classes must be able to provide a list of all shared resources under their management. Implementing classes must also be able to tell others if there are any resources under its management that are controlled by a given operations center. The shared resource manager is also responsible for periodically monitoring its shared resources to detect if the operations center controlling a resource doesn't have at least one user logged

into the system. When this condition is detected, the shared resource manager must push an event on the ResourceManagement event channel to notify others of this condition.

5.10.1.1.40SlotClipAudioData (Class)

This class is used to help keep track of and pass around slot data. This class associates a clip with a particular slot and usage, and with a file name which contains its audio (wav) data. The fileName is passed to the ISSAP55ProtocolHdlr to store the wav data in the slot.

5.10.1.1.41SyncCommandStatus (Class)

A SyncCommandStatus implements the CommandStatus interface and performs a notification when it is completed. It is used by the HAR service to track the activity of HARMessageNotifiers, which may operate asynchronously and provide status later via a CommandStatus.

5.10.1.1.42UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.10.1.1.43VoicePort (Class)

A voice port provides access to a port on a telephony board. It provides methods to connect it to a destination phone number and perform send and receive operations while connected that result in DTMF or voice being sent across the telephone connection to or from the device.

5.10.1.1.44VoicePortLocator (Class)

This class provides an implementation of the PortLocator's abstract connectPort() method that can connect a VoicePort that has been acquired by the PortLocator base class. This derived class logs information in the comm failure database table relating to connection problems that may occur. Since this is a telephony port which is much simpler to connect than, say, a ModemPort, there will be considerably fewer types of errors which can occur and thus be detected and reported.

5.10.2 Sequence Diagrams

5.10.2.1 HARControlModule:SetConfiguration (Sequence Diagram)

A user with the appropriate privileges can set the configuration of the HAR. The HAR must be in maintenance mode when setting the configuration. The command is processed asynchronously by the CommandQueue. When the command reaches the top of the queue, the HARImpl's setConfigurationImpl() method is called to do the actual work.

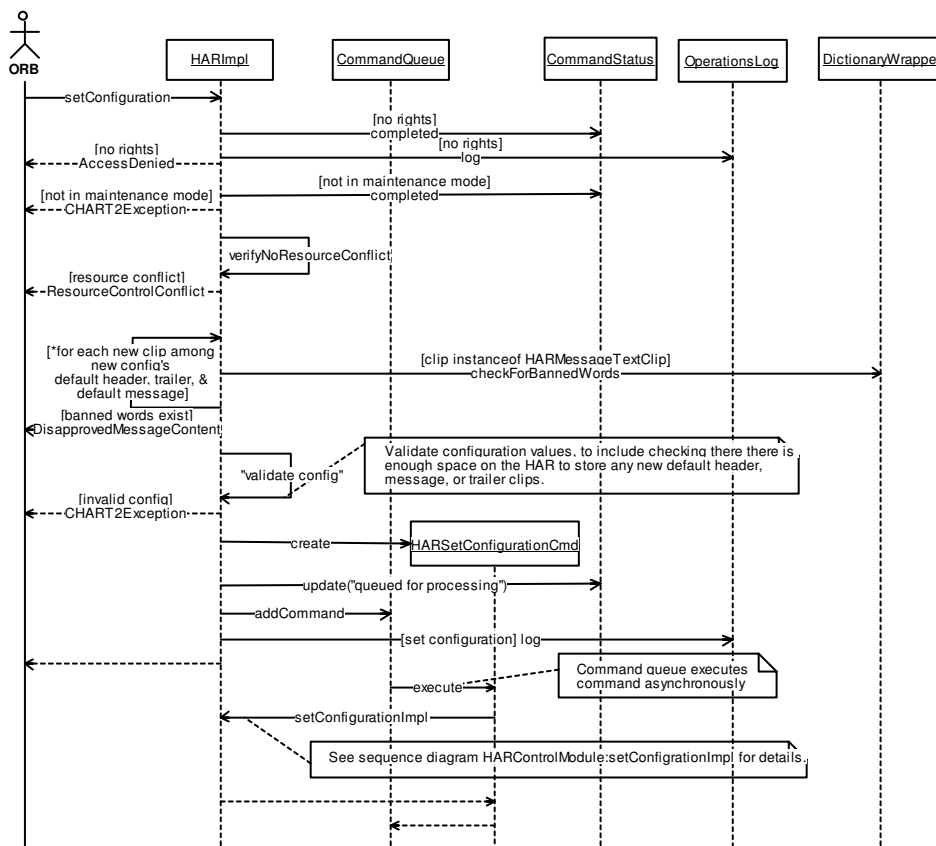


Figure 5-193 HARControlModule:SetConfiguration (Sequence Diagram)

5.10.2.2 HARControlModule:setConfigurationImpl (Sequence Diagram)

This method is called by the HARSetConfigCmd when it reaches the top of the CommandQueue and is executed. This method does the work of updating the configuration of the HAR. Some configuration elements require communication to the device: the default header, trailer, and message. If any of these change, the audio data is collected by calling prepareWavFiles() on each of the default clips changed, then a connected port is acquired and used to download the new clip data into the HAR. This is accomplished by calling the HARSlotManager store() method. Any clips which are unable to be stored are set back to their original values. Because the configuration consists of many separate values that are set individually on the device, the possibility of partial success exists. When this occurs warning messages are given back to the user through the command status object and the configuration is set to reflect the partial success. If any data has ultimately changed, the new configuration is stored and persisted, and a HARConfigurationChanged event is pushed.



5.11 INRIXDataImportModule

5.11.1 Classes

5.11.1.1 INRIXDataImportModuleClasses (Class Diagram)

This diagram shows the classes that make up the INRIXDataImportModule.

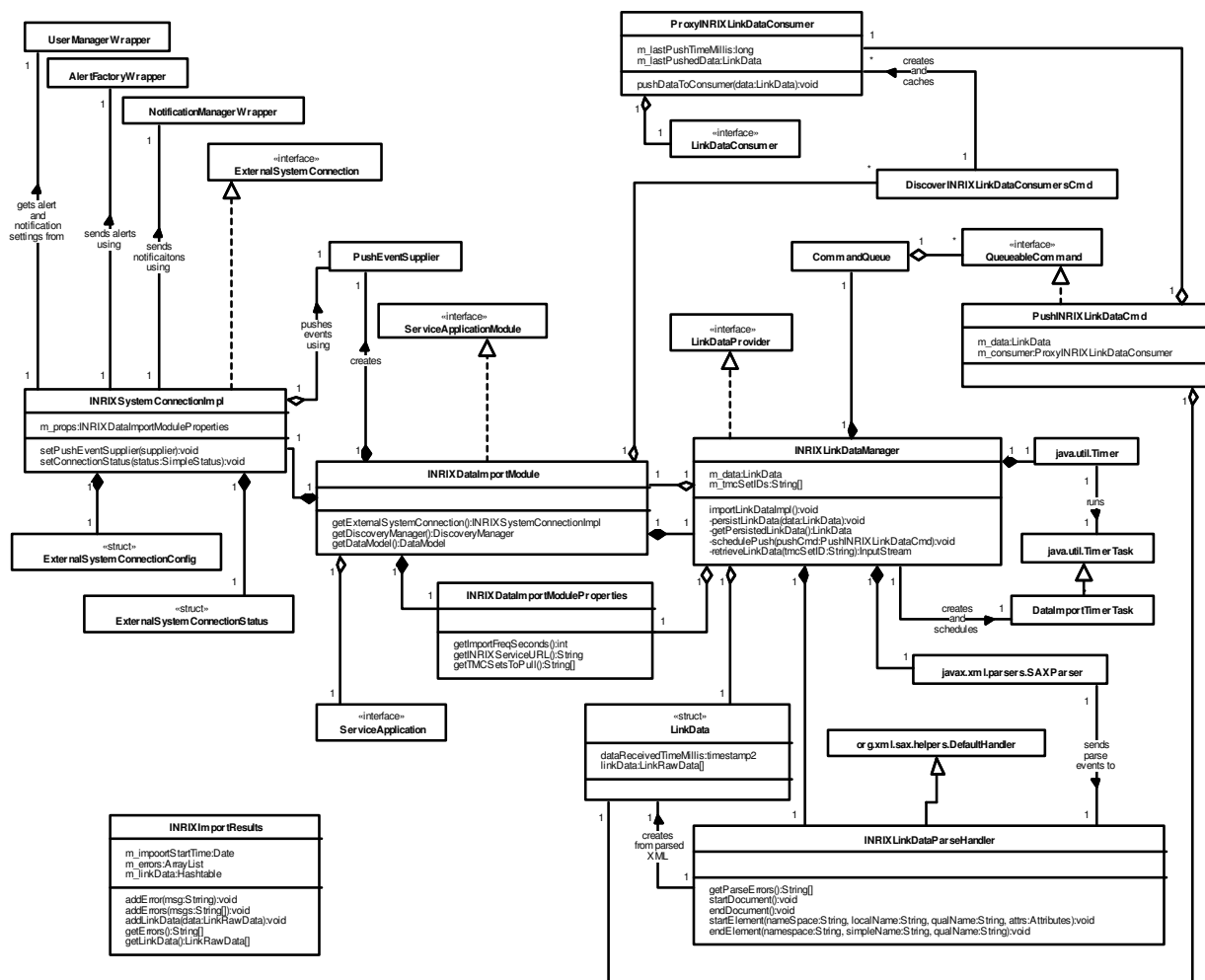


Figure 5-195 INRIXDataImportModuleClasses (Class Diagram)

5.11.1.1.1 AlertFactoryWrapper (Class)

This singleton class provides a wrapper for the Alert Factory that provides automatic

location of an Alert Factory and automatic re-discovery should the Alert Factory reference return an error. This class also allows for built-in fault tolerance by automatically failing over to a "working" Alert Factory without the user of this class being aware that this being done. In addition, this class defers the discovery of the Alert Factory until its first use, thus eliminating a start-up dependency for modules that use the Alert Factory.

This class delegates all of its method calls to the system AlertFactory using its currently known good reference to an AlertFactory. If the current reference returns a CORBA failure in the delegated call, this class automatically switches to another reference. When there are no good references (as is true the first time the object is used), this class issues a trader query to (re)discover the published Alert Factory objects in the system. During a method call, the trader will be queried at most one time and under normal circumstances, not at all.

5.11.1.1.2 CommandQueue (Class)

The CommandQueue class provides a queue for QueueableCommand objects. The CommandQueue has a thread that it uses to process each QueueableCommand in a first in first out order. As each command object is pulled off the queue by the CommandQueue's thread, the command object's execute method is called, at which time the command performs its intended task.

5.11.1.1.3 DataImportTimerTask (Class)

The TimerTask that is scheduled for periodic execution in order to import the latest data from the INRIX service.

5.11.1.1.4 DiscoverINRIXLinkDataConsumersCmd (Class)

5.11.1.1.5 ExternalSystemConnection (Class)

This interface defines external connections and provides status reporting.

5.11.1.1.6 ExternalSystemConnectionConfig (Class)

This struct defines a connection to an external system.

5.11.1.1.7 ExternalSystemConnectionStatus (Class)

This struct is used to report status for an ExternalSystemConnection.

5.11.1.1.8 INRIXDataImportModule (Class)

The service application module that provides INRIX data import functionality.

5.11.1.1.9 INRIXDataImportModuleProperties (Class)

This class provides convenience methods for getting the values of INRIX data import modules configuration properties.

5.11.1.1.10INRIXImportResults (Class)

This class is used to store information from the ingestion of INRIX link data. It stores errors encountered and link data.

5.11.1.1.11INRIXLinkDataManager (Class)

This class manages the current INRIXLinkData and provides access to it via the INRIXLinkDataProvider CORBA interface.

5.11.1.1.12INRIXLinkDataParseHandler (Class)

A SAX style parse handler that is used during the parse of the incoming INRIX XML document to create an INRIXLinkData structure for use by the application.

5.11.1.1.13INRIXSystemConnectionImpl (Class)

This class provides the implementation of the CORBA ExternalSystemConnection interface that represents the INRIX data connection. It is responsible for maintaining the connection status, pushing connection status events when the status changes and sending external connection related events and notifications.

5.11.1.1.14java.util.Timer (Class)

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

5.11.1.1.15java.util.TimerTask (Class)

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

5.11.1.1.16javax.xml.parsers.SAXParser (Class)

This class is used to perform a SAX parse of XML data.

5.11.1.1.17LinkData (Class)

This class contains the latest LinkRawData along with a timestamp that indicates when it was obtained by the CHART system.

5.11.1.1.18LinkDataConsumer (Class)

This CORBA interface defines the methods that a consumer of INRIX link data must implement in order to be updated when data changes.

5.11.1.1.19LinkDataProvider (Class)

This CORBA interface defines the methods that a provider of INRIX link data must implement.

5.11.1.1.20NotificationManagerWrapper (Class)

This singleton class presents the same interface as the NotificationManager, but uses a FirstAvailableOfferWrapper to provide fault tolerant access to the methods.

5.11.1.1.21org.xml.sax.helpers.DefaultHandler (Class)

This class provides a default base handler for a SAX parse of XML data.

5.11.1.1.22ProxyINRIXLinkDataConsumer (Class)

This class provides a proxy for an INRIXLinkDataConsumer. The proxy stores the last pushed data and the time it was last pushed to allow it to avoid pushing data that is not current.

5.11.1.1.23PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.11.1.1.24PushINRIXLinkDataCmd (Class)

This class provides an asynchronous method of pushing INRIXLinkData to a consumer.

5.11.1.1.25QueueableCommand (Class)

A QueueableCommand is an interface used to represent a command that can be placed on a CommandQueue for asynchronous execution. Derived classes implement the execute method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a CommandQueue. The CommandQueue driver calls the execute method to execute a command in the queue and a call to the interrupted method is made when a CommandQueue is shut down.

5.11.1.1.26ServiceApplication (Class)

This interface is implemented by objects that can provide the basic services needed by a ChartII service application. These services include providing access to basic CORBA objects that are needed by service applications, such as the ORB, POA, Trader, and Event

Service.

5.11.1.1.27ServiceApplicationModule (Class)

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

5.11.1.1.28UserManagerWrapper (Class)

The UserManagerWrapper is a singleton class that provides access to a single instance of a remote service type (in this case UserManager) where many instances may exist in the Traders. If the connection to the current instance is lost, it re-establishes the connection, possible with a different instance of the desired service type. This class supports storing properties with values that are string data or binary data.

5.11.2 SequenceDiagram

5.11.2.1 CHART2.INRIXDataImportModule:DataImportTimerTask.run (Sequence Diagram)

This diagram shows the processing that is performed each time the DataImportTimerTask is run. First an INRIXImportResults object is created to store all link data and errors from the import operation. Then an attempt is made to retrieve the link data for each configured TMC set. If the data cannot be obtained an error is added to the import results. If the data is retrieved successfully the returned XML document is parsed. All data found during the parse is added to the import results. When the parse has completed all parse errors are added to the import results as well. After all TMC sets have been imported the INRIXLinkDataManager will get the parsed data and any parse errors encountered from the INRIXImportResults object. If no data was found the connection status will be set to FAILURE. If data was found but errors were encountered the connection status will be set to WARNING. If all data was imported and no errors were encountered the connection status is set to OK. If there is any new data it is persisted and then pushed to all previously discovered ProxyINRIXDataConsumers asynchronously.

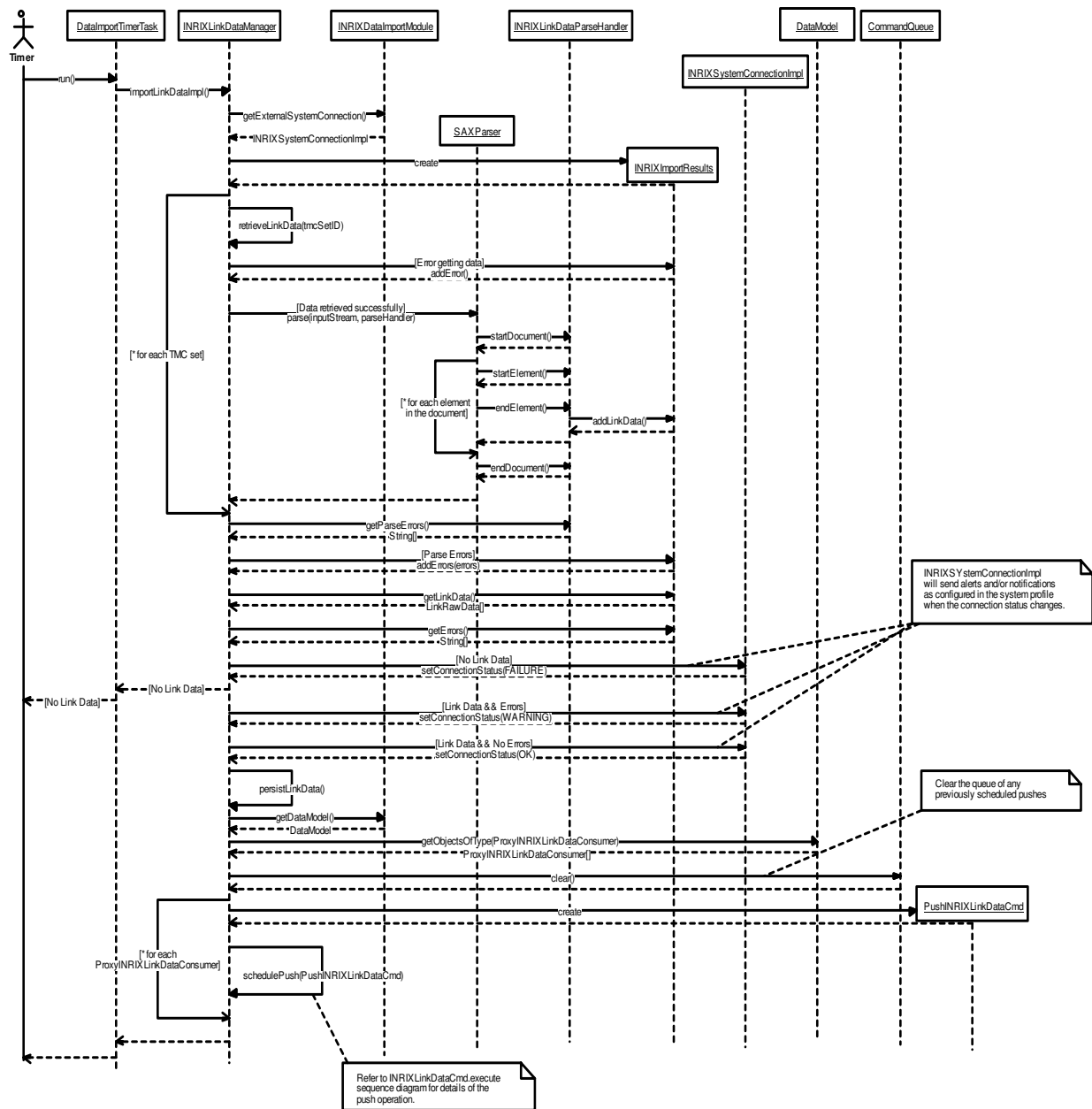


Figure 5-196 CHART2.INRIXDataImportModule:DataImportTimerTask.run (Sequence Diagram)

5.11.2.2 CHART2.INRIXDataImportModule:INRIXDataImportModule.initialize (Sequence Diagram)

This diagram shows the processing that is performed when the INRIXDataImportModule is initialized. The module creates a new INRIXDataImportModuleProperties object, then creates a new INRIXSystemConnectionImpl, activates it in the persistent POA and registers it in the primary trading service. Next the module creates a PushEventSupplier that will be used to push ExternalSystemConnection related events and registers the channel in the trader. The PushEventSupplier is set into the INRIXSystemConnectionImpl so that it may be used to push events as status changes. Next the INRIXLinkDataManager is created. The data manager reads any previously persisted link data to use as its initial link data and then creates a Timer and schedules a recurring DataImportTimerTask. The INRIXLinkDataManager also creates a

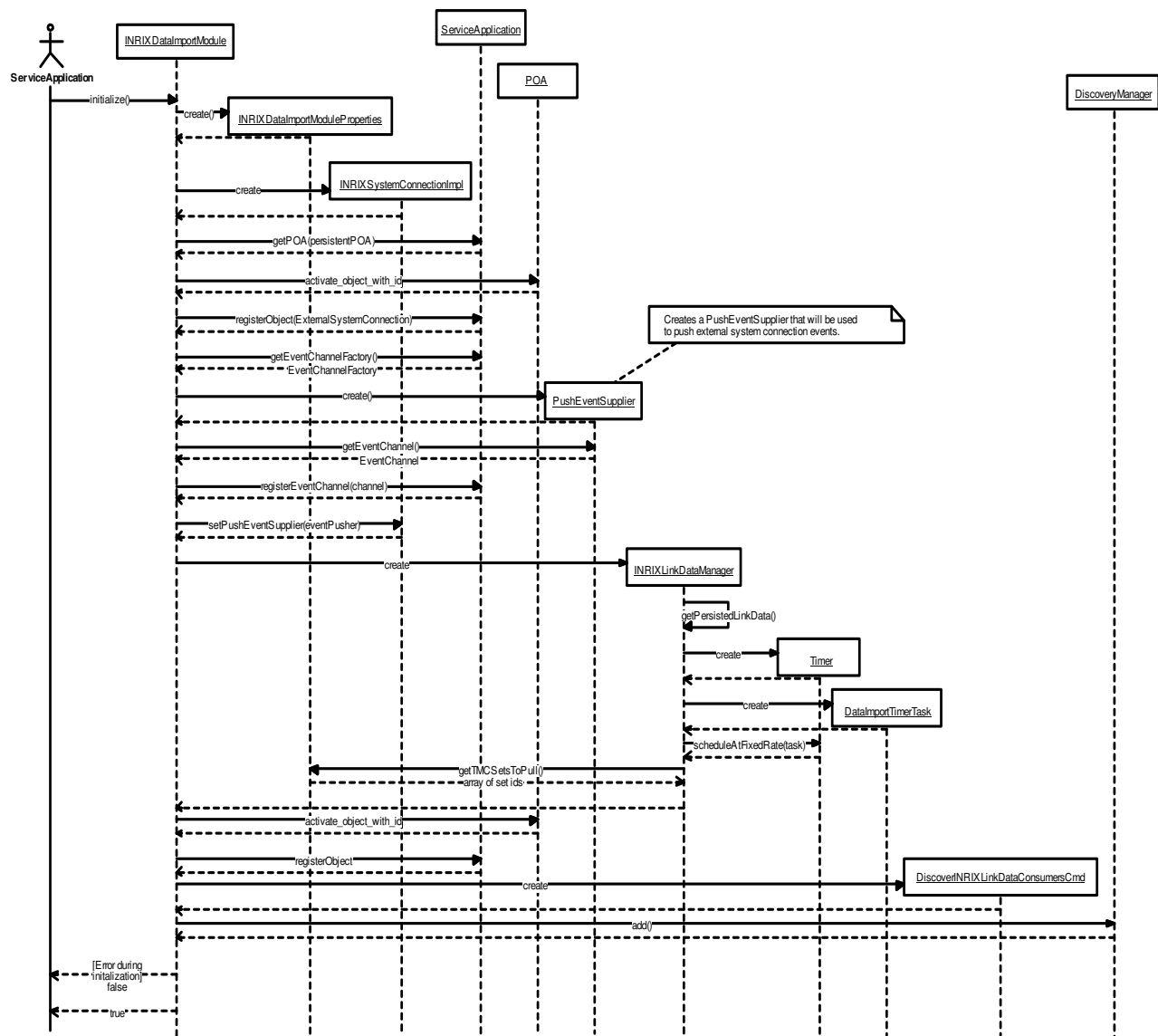


Figure 5-197 CHART2.INRIXDataImportModule:INRIXDataImportModule.initialize (Sequence Diagram)

5.11.2.3 CHART2.INRIXDataImportModule:INRIXLinkDataProvider.getLinkData (Sequence Diagram)

This diagram shows the processing that is performed each time a consumer calls the INRIXLinkDataProvider.getLinkData CORBA method. The data manager verifies that the caller has sufficient privileges to retrieve INRIX link data. If not an AccessDenied exception is thrown, otherwise the current link data is returned.

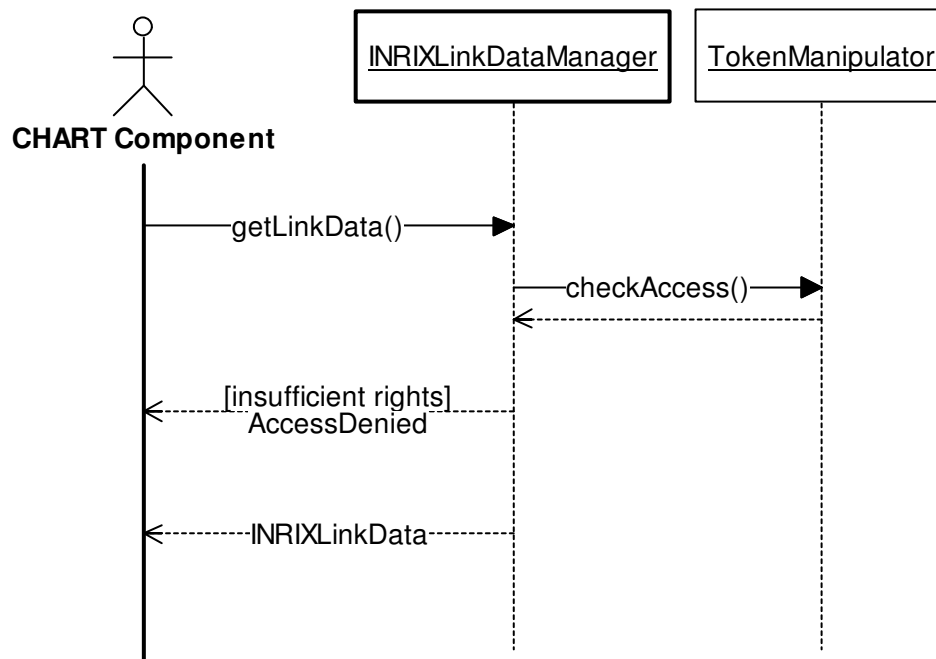


Figure 5-198 CHART2.INRIXDataImportModule:INRIXLinkDataProvider.getLinkData (Sequence Diagram)

5.11.2.4 CHART2.INRIXDataImportModule:PushINRIXLinkDataCmd.execute (Sequence Diagram)

This diagram shows the processing that is performed in order to push the current link data out to a INRIXLinkDataConsumer. The PushINRIXLinkDataCmd is executed by the CommandQueue and it calls the pushDataToConsumer method of the ProxyINRIXLinkDataConsumer. The proxy performs a check to verify that the data that is being pushed was received from INRIX more recently than the last data that was pushed to this consumer. If it is not the push is not performed. If it is, the INRIXLinkDataConsumer is called and is passed the new INRIXLinkData. If the consumer returns a result that it expected INRIX link data that was missing from the update the external system connection status is set to WARNING.

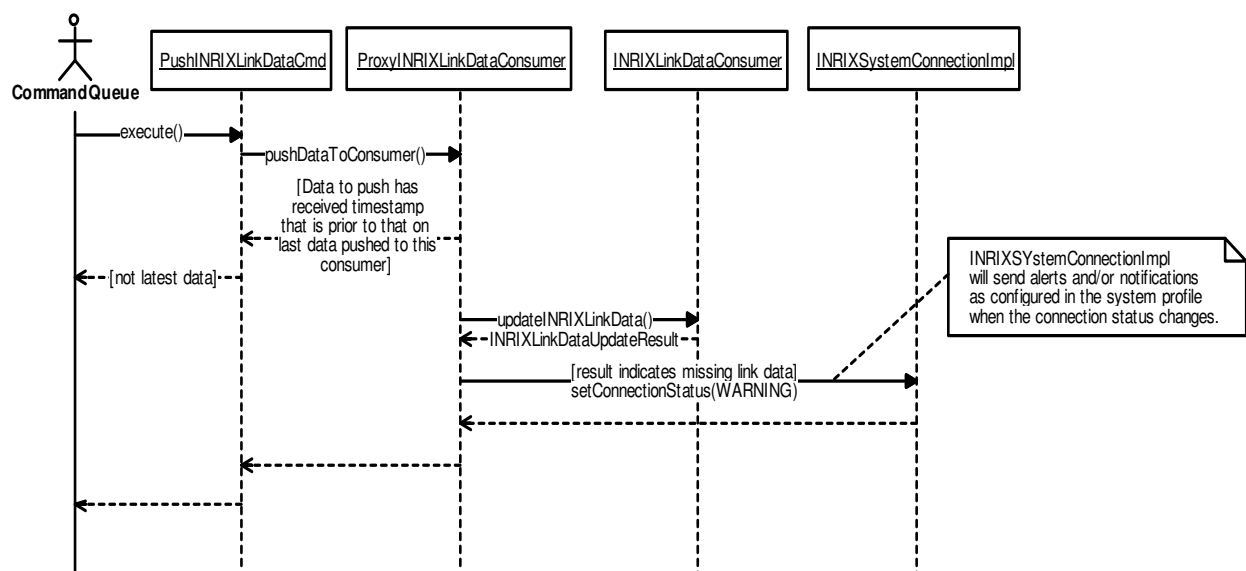


Figure 5-199 CHART2.INRIXDataImportModule:PushINRIXLinkDataCmd.execute (Sequence Diagram)

5.12 INRIXLinkImportProgramPkg

5.12.1 Classes

5.12.1.1 INRIXLinkDefImportProgram (Class Diagram)

Utility program used to import INRIX link definition data into the CHART system.

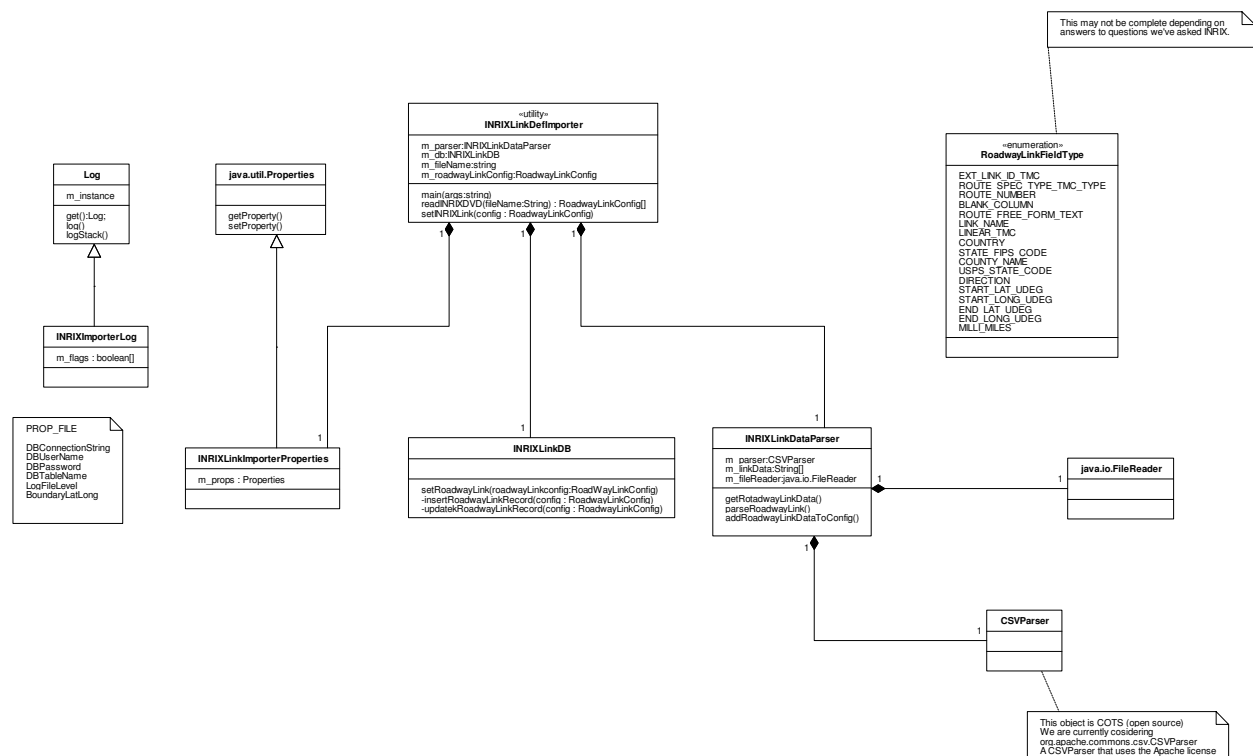


Figure 5-200 INRIXLinkDefImportProgram (Class Diagram)

5.12.1.1.1 CSVParser (Class)

This class is a COTS API that provides functionality for parsing CSV data.

5.12.1.1.2 INRIXImporterLog (Class)

The common logging class used by the INRIXLinkDefImportProgram to help reduce code duplication.

5.12.1.1.3 INRIXLinkDataParser (Class)

This class provides functions that reads the INRIX data line by line and converts that data into tokens, processes those tokens into data that is inserted into a RoadwayLocationConfig, and inserts or updates the configuration into the CHART database depending on whether

the link entry already exists in the database.

5.12.1.1.4 INRIXLinkDB (Class)

This class is a utility that provides methods for inserting or updating the database pertaining to INRIX link definition data.

5.12.1.1.5 INRIXLinkDefImporter (Class)

This is the main program of the INRIX link definition importer. It takes user parameters from the command line, reads program properties from the property file, controls reading the INRIX CSV link definition data file, converts the INRIX data and updates the CHART database.

5.12.1.1.6 INRIXLinkImporterProperties (Class)

This class provides a wrapper to the application's properties file that provides easy access to the properties specific to the INRIXLinkDefImportProgram. These properties include the name of the file where raw CSV INRIX Link definition data is located and the directory where debug log files are to be kept.

5.12.1.1.7 java.io.FileReader (Class)

Convenience class for reading character files.

5.12.1.1.8 java.util.Properties (Class)

The Properties class represents a persistent set of properties. The Properties can be saved to a stream or loaded from a stream. Each key and its corresponding value in the property list is a string. A property list can contain another property list as its "defaults"; this second property list is searched if the property key is not found in the original property list.

5.12.1.1.9 Log (Class)

Singleton log object to allow applications to easily create and utilize a LogFile object for system trace messages.

5.12.1.1.10 RoadwayLinkFieldType (Class)

Enumeration used as an index in to the array of tokens returned by the CSVParser. Also used to determine what process to run if any one each token from the array.

5.12.2 Sequence Diagrams

5.12.2.1 INRIXDefLinkImportProgramPkg:importINRIXLinks (Sequence Diagram)

This diagram shows how the INRIX Link definitions are imported in to the CHART database. The importer is a utility program executed from the command line. The user must provide a user name on the command line. The user can optionally provide the INRIX import data file name and table name on the command line. The program first gets the properties from the property file. It then checks to see if a user name was provided on the command line. If no user name is provided it loops a display to the user asking for one till one is provided. The program then creates a database file reader and uses it to create a CSVParser. Next it creates an INRIXLinkDB object. It then reads the data from the filename provided one line at a time and parses it into an array of tokens. It checks the lat/long to see if the link is in boundary specified by parameters from the prop file. If the link is not in the boundary it moves to the next line in the file. If the link is in the boundary the tokens are each processed as needed and stored in a RoadwayLinkConfig. The RoadwayLinkConfig is then added to the CHART database if it does not already exist in the database. It is updated if an entry already exists for that RoadwayLinkConfig. Processing continues at this point unless one of two situations occurs. If the first update/insert to the database fails an error is logged and displayed to the user and processing stops. If a property file parameter number of database errors occur the processing is also stopped. If the link configuration is successfully stored to the database the Link ID is logged in the application log and processing continues until all records have been processed from the INRIX link definition file

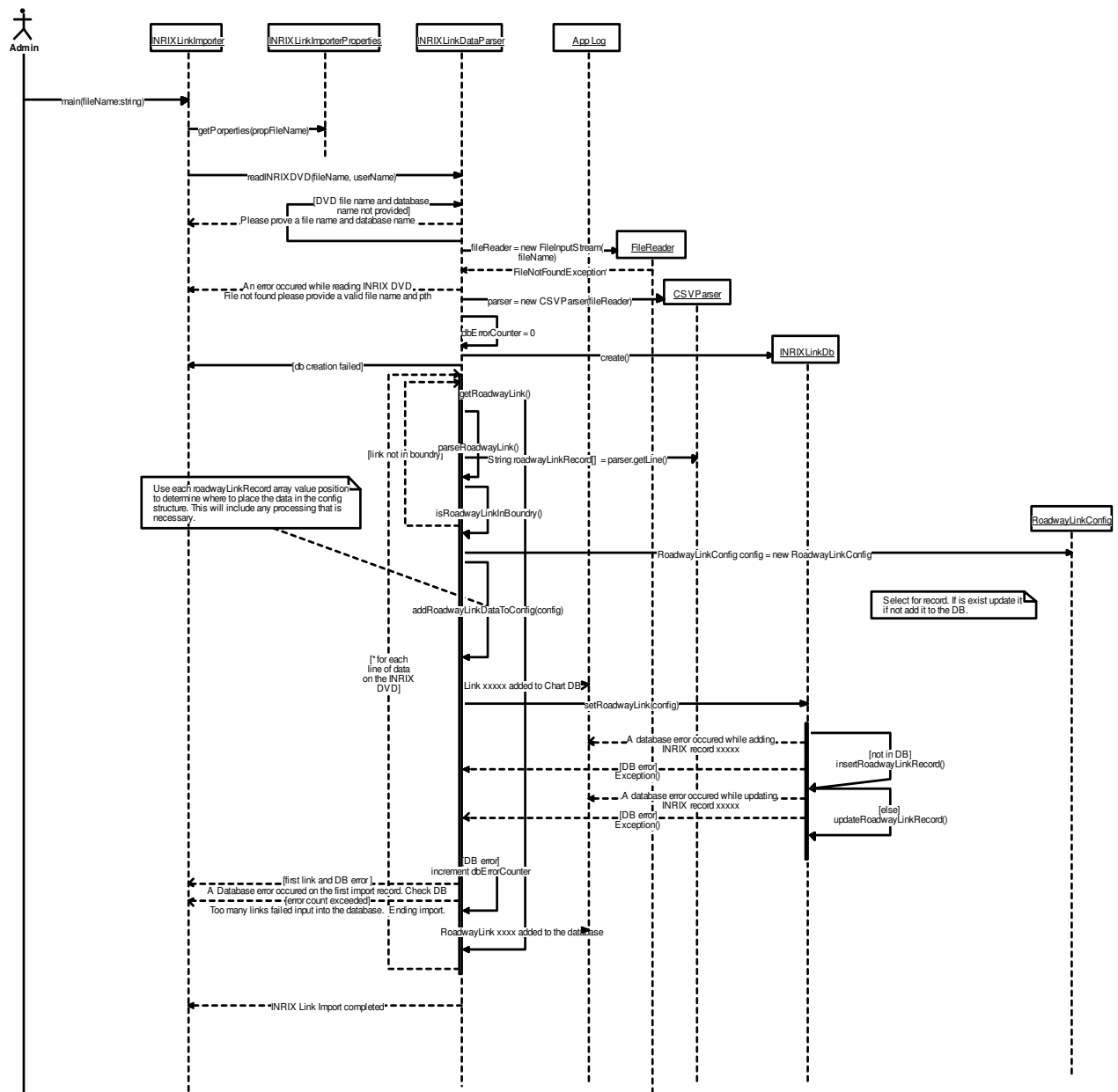


Figure 5-201 INRIXDefLinkImportProgramPkg:importINRIXLinks

5.13 Java Classes

5.13.1 Classes

5.13.1.1 JavaClasses (Class Diagram)

This package is included for reference to classes included in the Java programming language that are used in class and sequence diagrams for other packages within this design.

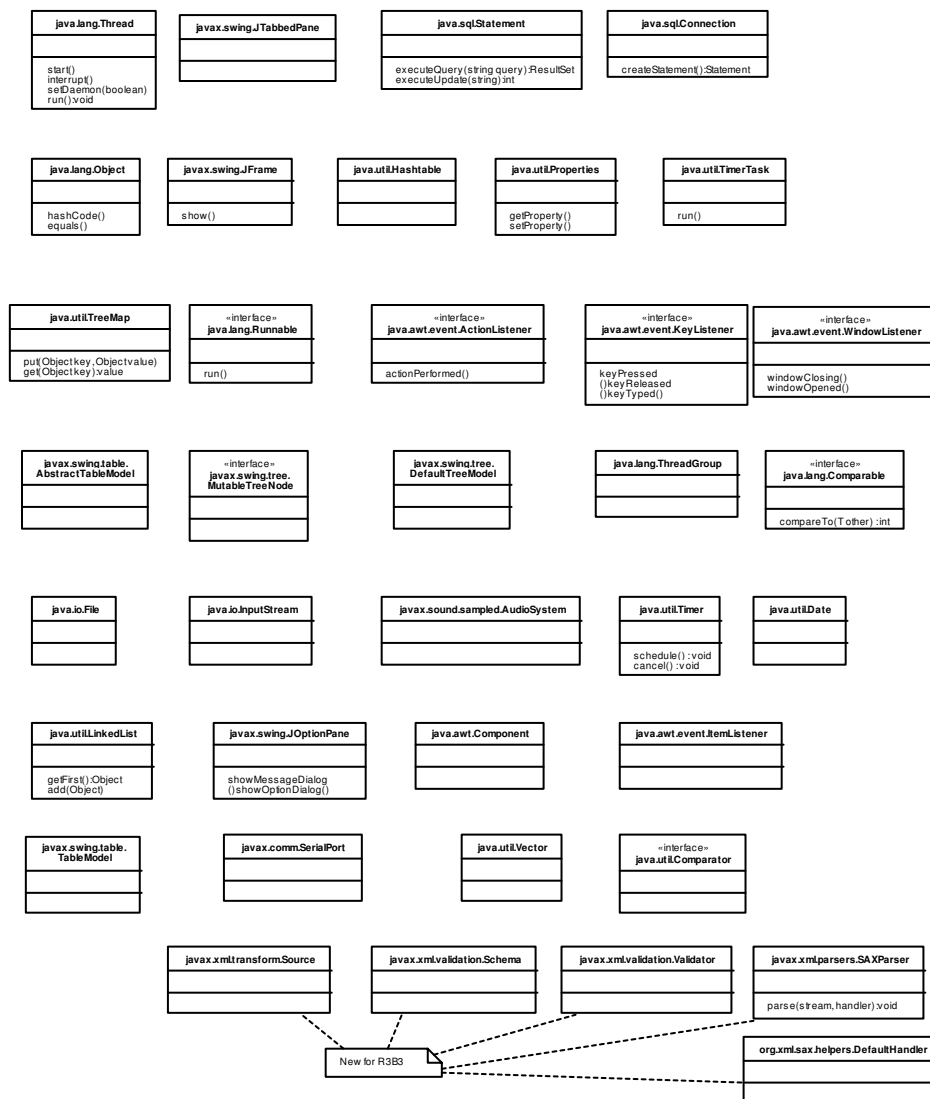


Figure 5-202 JavaClasses (Class Diagram)

5.13.1.1.1 java.awt.Component (Class)

This class is the base class for all graphical user interface components such as buttons and panels.

5.13.1.1.2 java.awt.event.ActionListener (Class)

This interface listens for actions such as when a menu item or button is clicked. For menu items, it is attached to menu items when the menu is built.

5.13.1.1.3 java.awt.event.ItemListener (Class)

This interface allows the implementing class to listen for changes to an item such as a list item or combo box item.

5.13.1.1.4 java.awt.event.KeyListener (Class)

Interface that a class must realize in order for objects of that class to be notified when the user presses a key.

5.13.1.1.5 java.awt.event.WindowListener (Class)

Listener interface that a class must implement for receiving window events

5.13.1.1.6 java.io.File (Class)

This class is an abstract representation of file and directory pathnames.

5.13.1.1.7 java.io.InputStream (Class)

Java class that represents a input stream of bytes.

5.13.1.1.8 java.lang.Comparable (Class)

This interface allows two objects to be compared for the purposes of sorting.

5.13.1.1.9 java.lang.Object (Class)

This is the base class from which all Java classes inherit.

5.13.1.1.10 java.lang.Runnable (Class)

This interface allows the run method to be called from another thread using Java's threading mechanism.

5.13.1.1.11 java.lang.Thread (Class)

This class represents a java thread of execution.

5.13.1.1.12 java.lang.ThreadGroup (Class)

A thread group represents a set of threads.

5.13.1.1.13 java.sql.Connection (Class)

This class represents a connection (session) with a specific database.

5.13.1.1.14 java.sql.Statement (Class)

Java class used for executing a static SQL statement and obtaining the results produced by it.

5.13.1.1.15 java.util.Comparator (Class)

This interface is implemented by classes that can be sorted.

5.13.1.1.16 java.util.Date (Class)

A class used to store dates and times.

5.13.1.1.17 java.util.Hashtable (Class)

This class implements a hashtable, which is a data structure that maps keys to values. Any non-null object can be used as a key or as a value. Objects used as keys implement the hashCode method which is inherited by all objects from the java.lang.Object class.

5.13.1.1.18 java.util.LinkedList (Class)

This class is an implementation of List interface for a linked list.

5.13.1.1.19 java.util.Properties (Class)

The Properties class represents a persistent set of properties. The Properties can be saved to a stream or loaded from a stream. Each key and its corresponding value in the property list is a string. A property list can contain another property list as its "defaults"; this second property list is searched if the property key is not found in the original property list.

5.13.1.1.20 java.util.Timer (Class)

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

5.13.1.1.21 java.util.TimerTask (Class)

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

5.13.1.1.22 java.util.TreeMap (Class)

This class is an implementation of the SortedMap interface. This class guarantees that the

map will be in ascending key order, sorted according to the natural order for the key's class, or by the comparator provided at creation time, depending on which constructor is used.

5.13.1.1.23 java.util.Vector (Class)

A Vector is a growable array of objects.

5.13.1.1.24 javax.comm.SerialPort (Class)

This class provides access to a computer's serial port. It allows the port to be opened and closed and allows data to be sent and received.

5.13.1.1.25 javax.sound.sampled.AudioSystem (Class)

The AudioSystem class acts as the entry point to the sampled-audio system resources. This class lets you query and access the mixers that are installed on the system.

5.13.1.1.26 javax.swing.JFrame (Class)

Java class that displays a frame window.

5.13.1.1.27 javax.swing.JOptionPane (Class)

This class is used to display popup messages to an end user.

5.13.1.1.28 javax.swing.JTabbedPane (Class)

This class is a component that has tabbed pages, and the user can click on a tab to flip to a certain page.

5.13.1.1.29 javax.swing.table. AbstractTableModel (Class)

This class provides a base implementation of the TableModel interface. This data structure will be used to supply a JTable with data.

5.13.1.1.30 javax.swing.table. TableModel (Class)

This class provides the data structure that drives the population and updating of the data used by the JTable (a Java GUI component).

5.13.1.1.31 javax.swing.tree. DefaultTreeModel (Class)

This class is the data structure which is used as a foundation for the JTree class.

5.13.1.1.32 javax.swing.tree. MutableTreeNode (Class)

This interface extends the TreeNode interface and provides the ability to add and remove children from nodes. It may be used in a TreeModel.

5.13.1.1.33javax.xml.parsers.SAXParser (Class)

This class is used to perform a SAX parse of XML data.

5.13.1.1.34javax.xml.transform.Source (Class)

This class is used for XML schema validation.

5.13.1.1.35javax.xml.validation.Schema (Class)

This class represents an XSD schema for XML.

5.13.1.1.36javax.xml.validation.Validator (Class)

This class represents a validator that can validate XML against an XSD schema.

5.13.1.1.37org.xml.sax.helpers.DefaultHandler (Class)

This class provides a default base handler for a SAX parse of XML data.

5.14.1.1.1 DMSTravInfoMsgTemplate (Class)

The DMSTravInfoMsgTemplate interface is implemented by objects that allow execution of tasks associated with DMS travel information message templates.

5.14.1.1.2 DMSTravInfoMsgTemplateConfig (Class)

This object contains the configuration data for a message template that represents a DMSTravInfoMsgTemplate in the CHART DB

5.14.1.1.3 DMSTravInfoMsgTemplateConfigInfo (Class)

This structure contains a DMSTravInfoMsgTemplateConfig and a time stamp of when the configuration was retrieved from the database. It's used in conjunction with a property to determine if it is time to read the configuration from the database again. This evaluation occurs when the template configuration is requested.

5.14.1.1.4 DMSTravInfoMsgTemplateDB (Class)

This class is a utility that provides methods for adding, removing, and updating database data pertaining to DMS Travel Information Message Templates.

5.14.1.1.5 DMSTravInfoMsgTemplateImpl (Class)

This class implements the DMSTravInfoMsgTemplate interface defined in the IDL that provides the set of methods use to create, view and remove DMS travel info message templates

5.14.1.1.6 java.util.Properties (Class)

The Properties class represents a persistent set of properties. The Properties can be saved to a stream or loaded from a stream. Each key and its corresponding value in the property list is a string. A property list can contain another property list as its "defaults"; this second property list is searched if the property key is not found in the original property list.

5.14.1.1.7 MessageTemplateFactory (Class)

Interface whose implementation is used to create message templates, retrieve travel information message templates and retrieve travel time and toll rate formats.

5.14.1.1.8 MessageTemplateFactoryImpl (Class)

This class is an implementation of MessageTemplateFactory and is capable of creating a new DMSTravInfoMsgTemplate objects in the system. Additionally, it provides the services of retrieving existing travel information message formats and DMSTravInfoMsgTemplates.

5.14.1.1.9 MessageTemplateFormats (Class)

This structure contains all travel time and toll rate format that are specified within a given

DMSTravInfoMsgTemplate.

5.14.1.1.10 MessageTemplateModule (Class)

This class provides the resources and support functionality necessary to serve DMSTravInfoMsgTemplates related objects in a service application. It implements the ServiceApplicationModule interface which allows it to be served from any ServiceApplication.

5.14.1.1.11 MessageTemplateModuleProperties (Class)

This class provides a wrapper to the application's properties file that provides easy access to the properties specific to the MessageTemplateModule. These properties include the name of the file where raw DMS Message Template parameter data is to be logged, the directory where debug log files are to be kept, and the interval at which the configuration of message formats and DMSTravInfoMsgTemplates is to be read from the DB when requested.

5.14.1.1.12 PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.14.1.1.13 ServiceApplicationModule (Class)

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

.

5.14.2 Sequence Diagrams

5.14.2.1 DMSTravInfoMsgTemplateImpl:getConfig (Sequence Diagram)

This sequence diagram shows the process for getting a DMSTravInfoMsgTemplateConfig from an DMSTravInfoMsgTemplateImpl. The function first checks to see if the caller has the correct access rights. If the caller does not have the proper rights the function throws an AccessDenied exception. If the caller has the correct rights the process continues and next checks to see if the configuration has been cached longer than the TemplateCacheThreshold specifics. If the threshold has not been exceeded it returns the cached DMSTravInfoMsgTemplateConfig configuration. If the TemplateCacheThreshold has been exceeded the function reads the configuration using the DMSTravInfoMsgTemplateDb, from the DMS_TRAVEL_INFO_MSGE_TEMPLATE table and the various MSG_FORMATS tables. It then returns the DMSTravInfoMsgTemplateConfig to the caller.

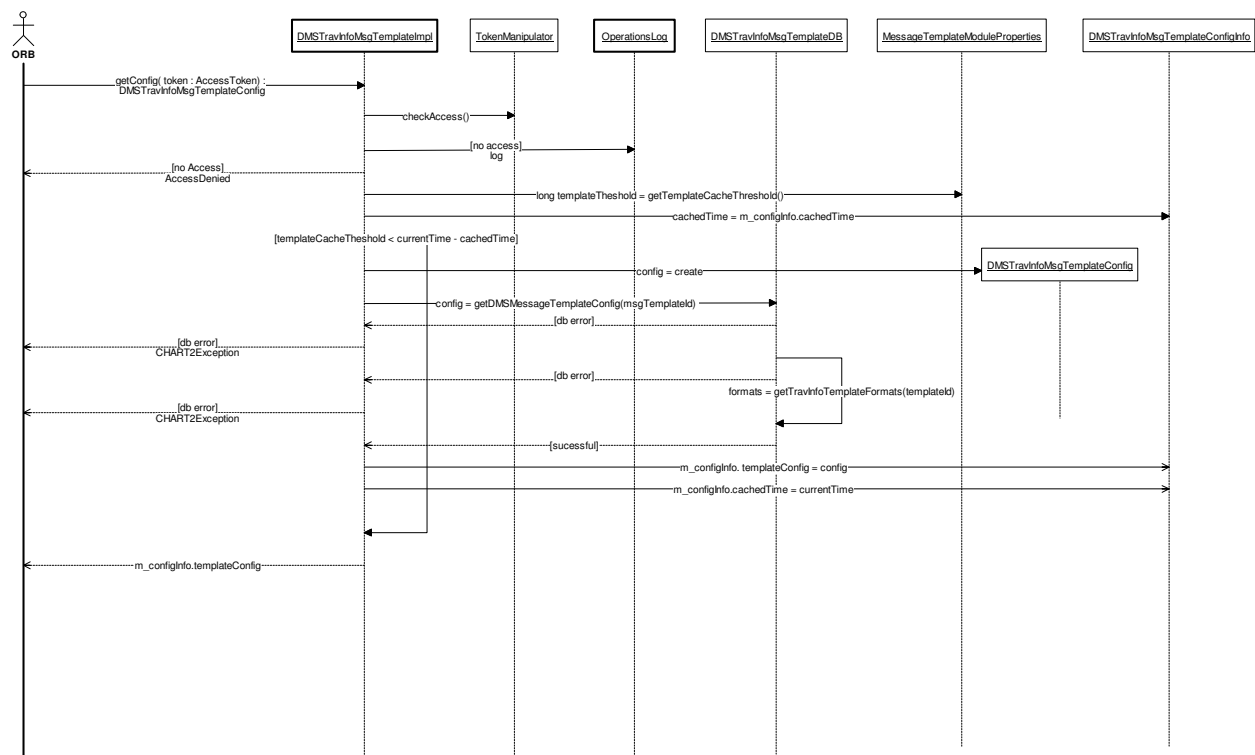


Figure 5-204. DMSTravInfoMsgTemplateImpl::getConfig (Sequence Diagram)

5.14.2.2 DMSTravInfoMsgTemplateImpl:remove (Sequence Diagram)

This sequence diagram describes the process of removing a DMS message travel info template from the CHART system.

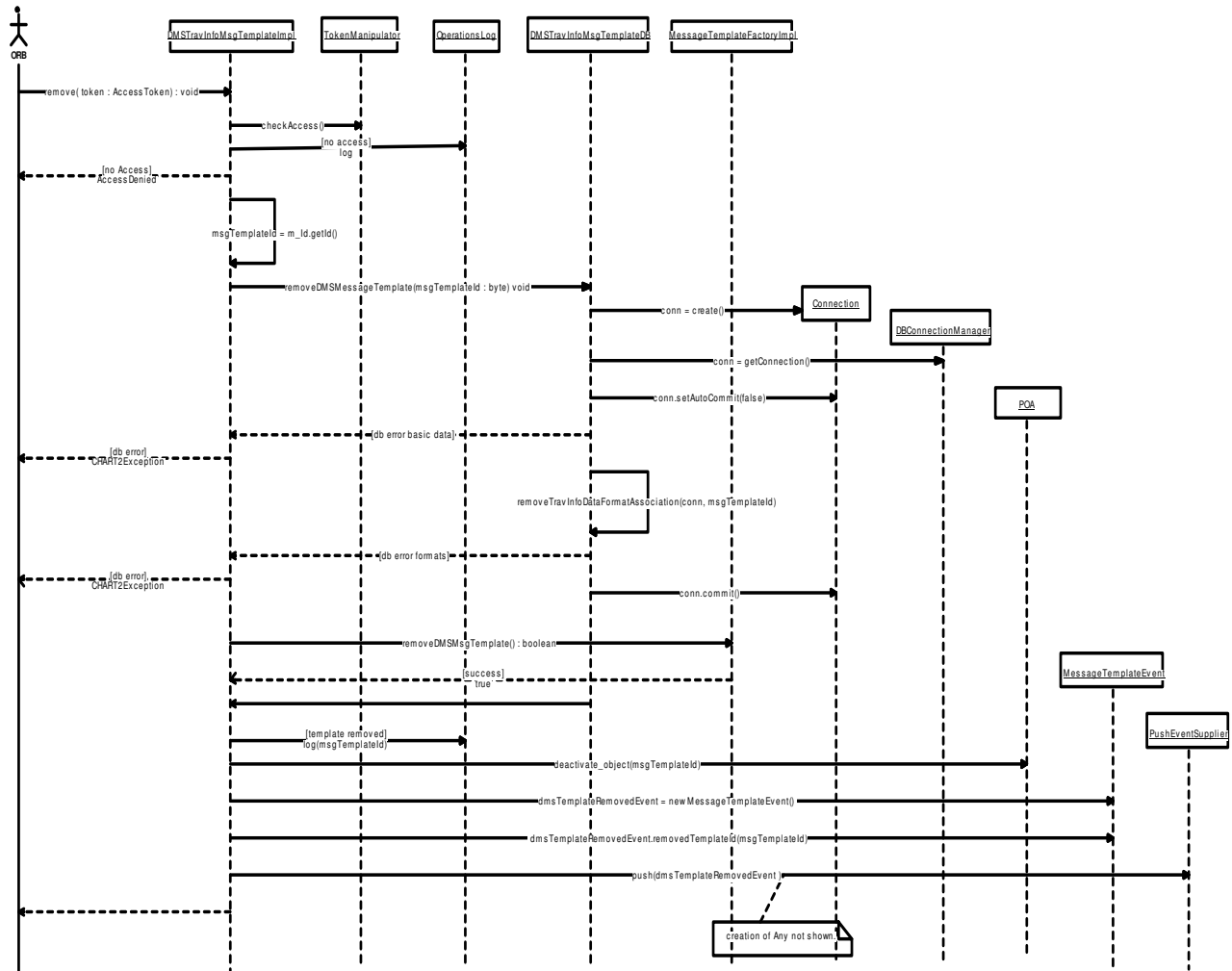


Figure 5-205. DMSTravInfoMsgTemplateImpl:remove (Sequence Diagram)

5.14.2.3 DMSTravInfoMsgTemplateImpl:setConfig (Sequence Diagram)

This method is when a user updates the configuration of a DMSTravInfoMsgTemplate. The method first checks the user's access rights. If the user has the proper rights the method saves the configuration to the database by calling addDMSMessageTemplateConfig on DMSTravInfoMsgTemplateDB. addDMSMessageTemplateConfig creates a database connection with auto commit set to false. It then saves the configuration base data to the database using updateConfigData(). If there is a database error the changes are rolled back and a db error is returned to the DMSTravInfoMsgTemplateImpl. The impl throws a Chart2Exception. addDMSMessageTemplateConfig calls the setTravInfoTemplateFormats() to update the Message Template Formats. Errors are handled the same as in updateConfigData(). If the database inserts/updates are successful the changes are committed, a DMSTravInfoMsgTemplateInfo is created and a dmsTemplateChangedEvent is push to clients.

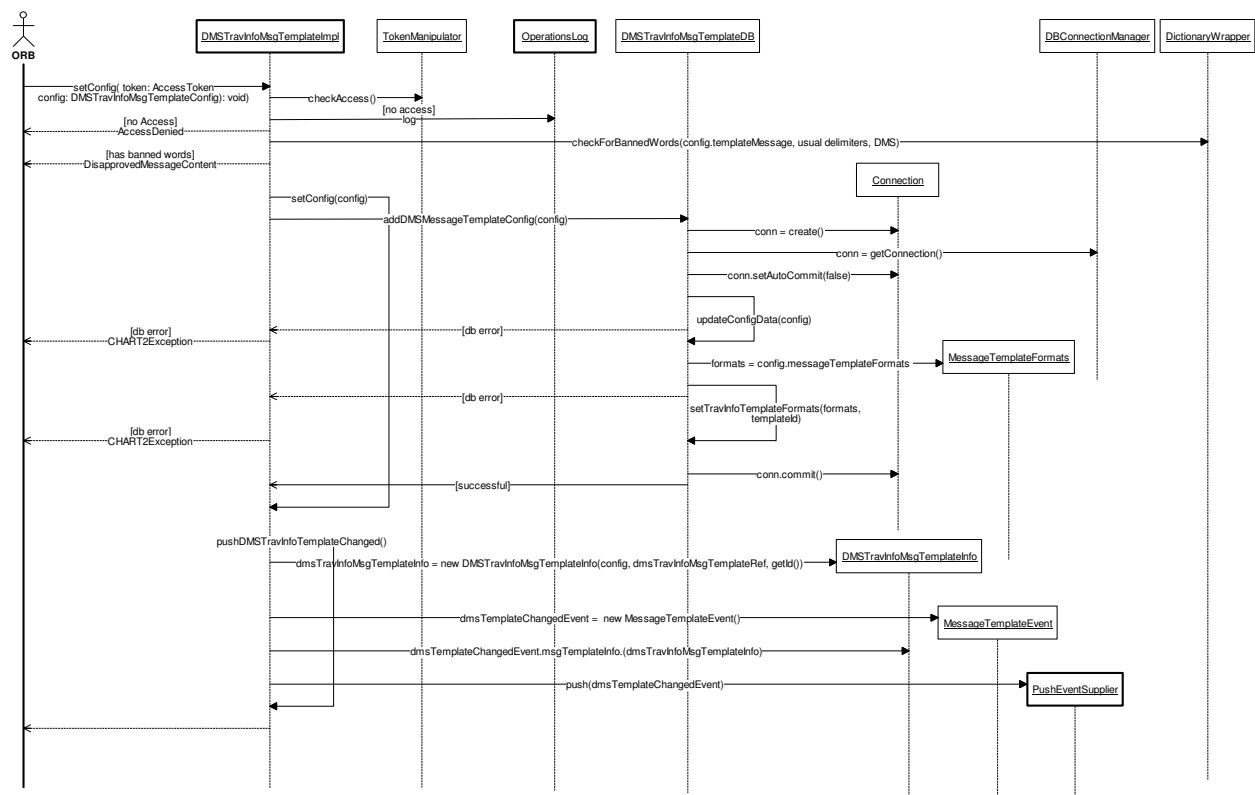


Figure 5-206. DMSTravInfoMsgTemplateImpl:setConfig (Sequence Diagram)

5.14.2.4 MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate (Sequence Diagram)

This diagram shows how a new DMSTravInfoMsgTemplate is created.

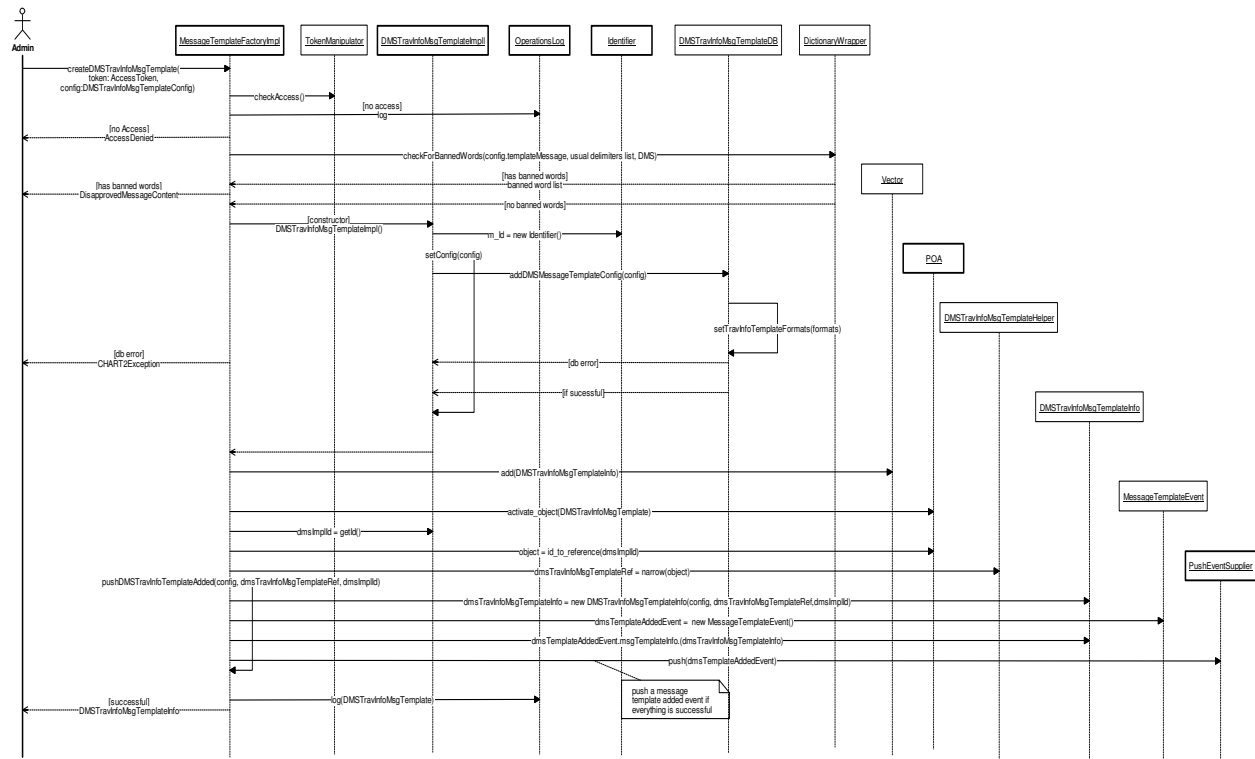


Figure 5-207. MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate (Sequence Diagram)

5.14.2.5 MessageTemplateFactoryImpl:getDMSTravInfoMsgTemplates (Sequence Diagram)

This diagram shows how all DMS travel information messages templates in the CHART system are retrieved..

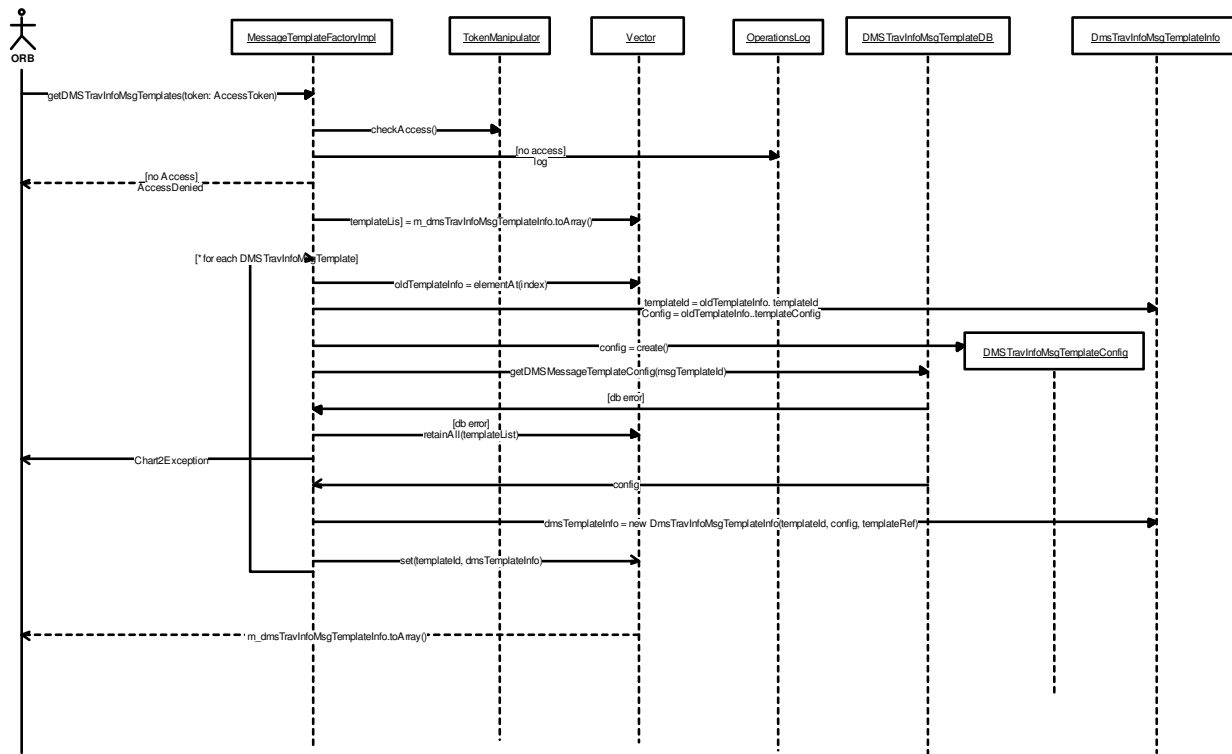


Figure 5-208. MessageTemplateFactoryImpl:getDMSTravInfoMsgTemplates (Sequence Diagram)

5.14.2.6 MessageTemplateFactoryImpl:getTollRateTimeFormats (Sequence Diagram)

This sequence diagram shows the getTollRateTimeFormats function. It first checks to see if the caller has the proper access rights. If not it throws an AccessDenied exception and stops operations. If the user has all rights needed the function checks if the formatThreshold has been exceeded by comparing it to the cached formats time stamp. If not exceeded it returns the currently cached toll rate time formats. If the threshold has been exceeded the function reads the formats from the database and returns them to the requester. This diagram serves as the example for all other MessageTemplateFactoryImpl get format functions. getTravelTimeFormats(), getTravelTimeRangeFormats(), getTollRateFormats() and getDistanceFormats() are essentially the same functionally as getTollRateTimeFormats().

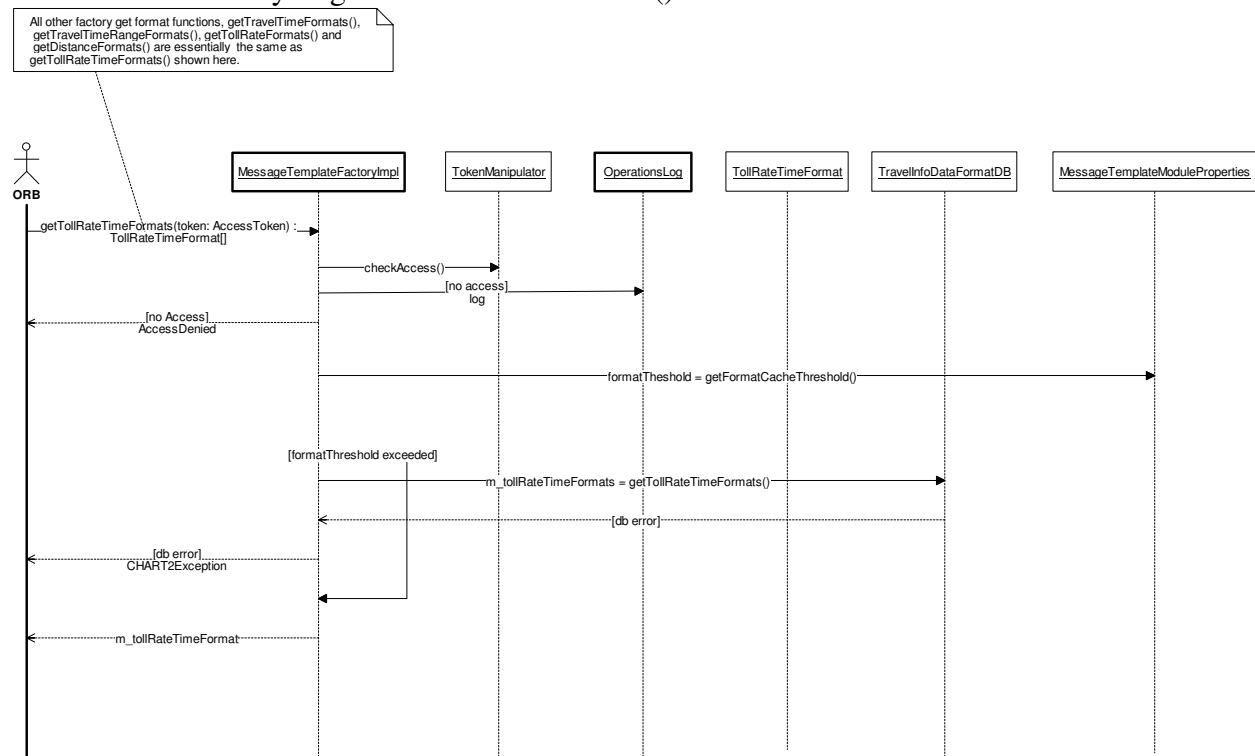


Figure 5-209. MessageTemplateFactoryImpl:getTollRateTimeFormats (Sequence Diagram)

5.14.2.7 MessageTemplateModule:initialize (Sequence Diagram)

This diagram shows what happens when the MessageTemplateModule is initialized. The ServiceApplication calls the MessageTemplateModule to initialize, which reads in the properties from a file, overriding the default properties. It creates an event channel for message templates and publishes the channel in the trading service so that other applications can see it. It creates a DMSTravInfoMsgTemplateDB object to handle all of the database calls, and a MessageTemplateFactoryImpl object to manage the DMS travel info message templates. The MessageTemplateFactoryImpl calls the DMSTravInfoMsgTemplateDB to load the DMSTravInfoMsgTemplate objects from the database. Then for each template it will activate the an existingDMSTravInfoMsgTemplate objects. The MessageTemplateFactoryImpl is exported to the trading service.

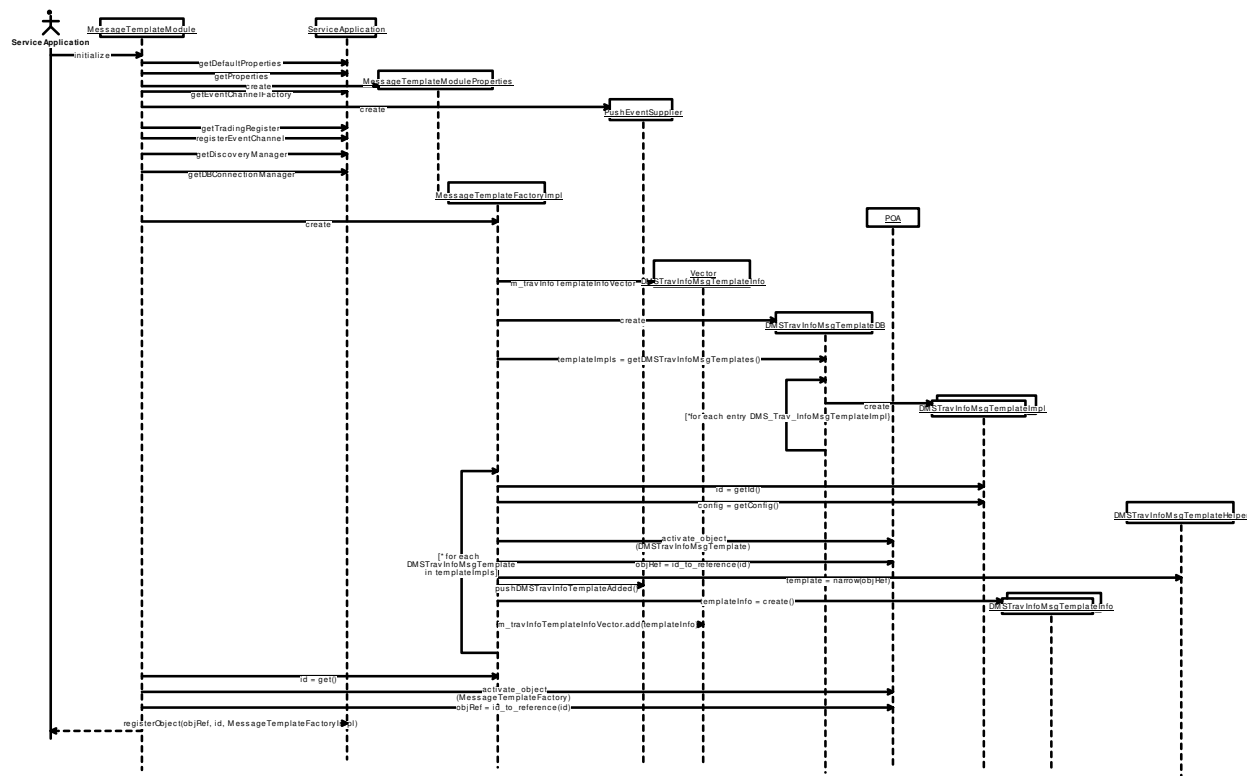


Figure 5-210. MessageTemplateModule:initialize (Sequence Diagram)

5.14.2.8 MessageTemplateModule:shutdown (Sequence Diagram)

This diagram shows what happens when the MessageTemplateModule shuts down.

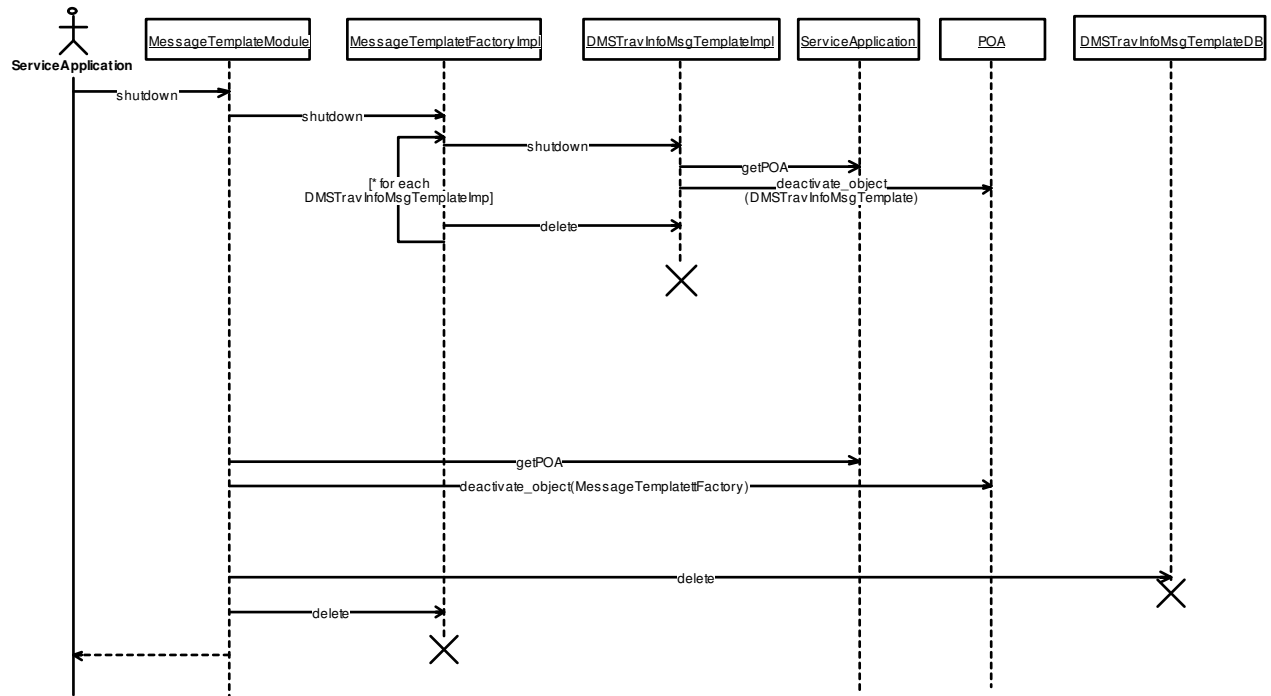


Figure 5-211. MessageTemplateModule:shutdown (Sequence Diagram)

5.15 RoadwayLocationModule

5.15.1 Classes

5.15.1.1 RoadwayLocationModule (Class Diagram)

This diagram shows the classes involved in the implementation of the Roadway Location Lookup Module. The Roadway Location Lookup Module is used to look up specific roadway location attributes within the State of Maryland, for instance, a list of Maryland counties, a list of primary roadways within a Maryland County, or a list of intersecting roadways along a given primary roadway within a given Maryland County.

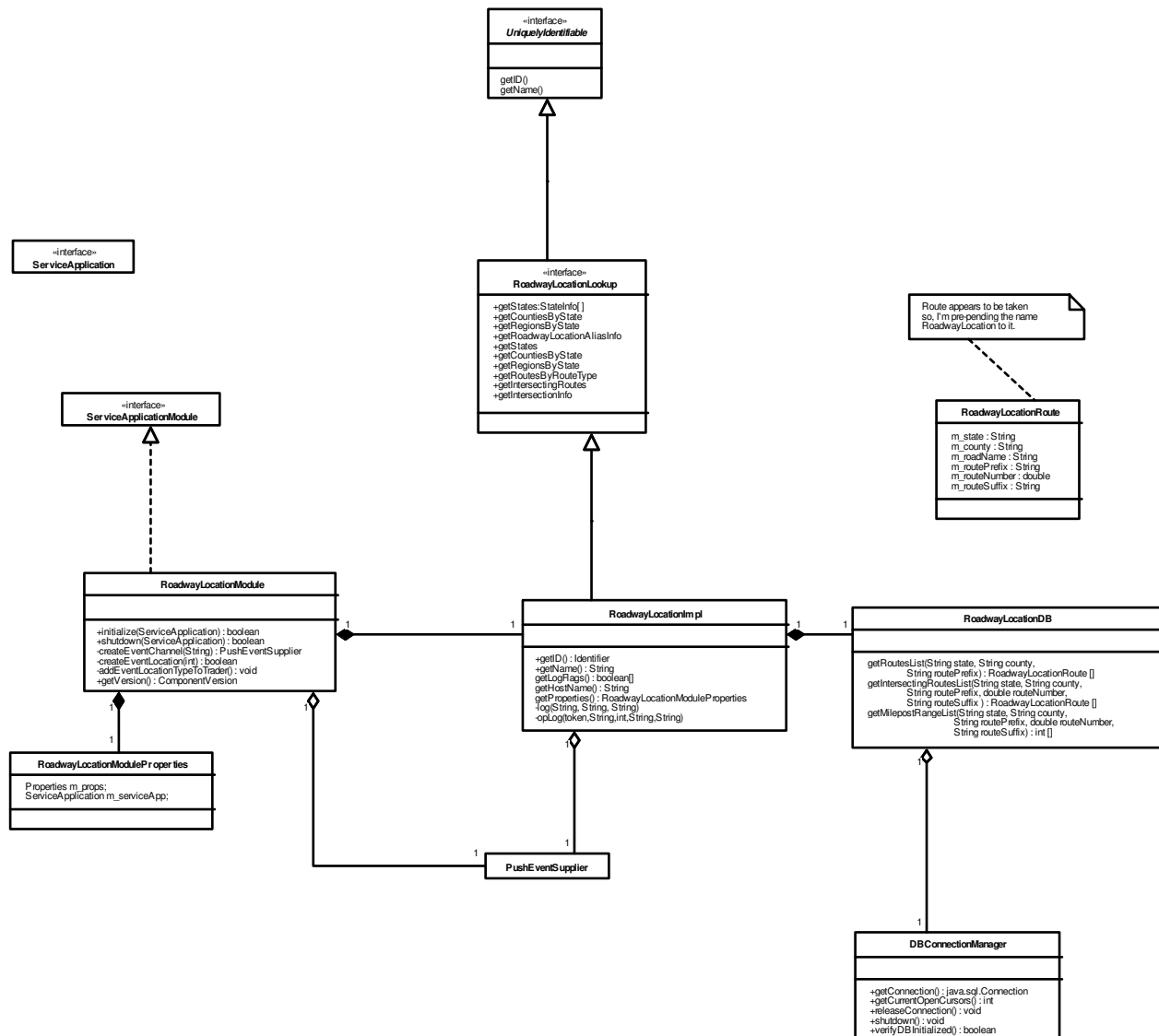


Figure 5-212. RoadwayLocationModule (Class Diagram)

5.15.1.1.1 DBConnectionManager (Class)

This class implements a database connection manager that manages a pool of database connections. Any CHART II system thread requiring database access gets a database connection from the pool of connections maintained by this manager class. The connections are maintained in two separate lists namely, inUseList and freeList. The inUseList contains connections that have already been assigned to a thread. The freeList contains unassigned connections. This class assumes that an appropriate JDBC driver has been loaded either by using the "jdbc.drivers" system property or by loading it explicitly. The class has a monitor thread that is started by the constructor. This connection monitor thread periodically checks the inuseList to see if there are connections that are owned by dead threads and move such connections to the freeList. The connection monitor thread is started only if a non-zero value is specified for the monitoring time interval in the constructor.

5.15.1.1.2 PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.15.1.1.3 RoadwayLocationDB (Class)

The RoadwayLocationDB class provides an interface between the RoadwayLocation service and the GIS database. It contains a collection of methods that perform database operations on tables pertinent to RoadwayLocation. The class is constructed with a DBConnectionManager object, which manages database connections.

5.15.1.1.4 RoadwayLocationImpl (Class)

The RoadwayLocationImpl class provides an implementation of the RoadwayLocation interface.

The RoadwayLocationImpl contains *Impl methods that map to methods specified in the IDL. In release R3B1, the methods are an interface to the GIS mapping database that will be used primarily by CHARTLite to get Roadway Location information.

5.15.1.1.5 RoadwayLocationLookup (Class)

This class is used to actually do the lookups in the RoadwayLocationDB.

5.15.1.1.6 RoadwayLocationModule (Class)

The RoadwayLocationModule class is the service module for Roadway Location. It implements the ServiceApplicationModule interface. It creates and serves a single RoadwayLocationImpl object. It also creates RoadwayLocationDB, RoadwayLocationModuleProperties, and PushEventSupplier objects.

5.15.1.1.7 RoadwayLocationModuleProperties (Class)

The RoadwayLocationModuleProperties class is used to provide access to properties used by the Roadway Location Module. This class wraps properties that are passed to it upon construction. It adds its own defaults and provides methods to extract properties specific to the Roadway Location Module.

5.15.1.1.8 RoadwayLocationRoute (Class)

The RoadwayLocationRoute class is a data structure that will hold the information about the roadway location, such as state, county, road name, route number, etc.

5.15.1.1.9 ServiceApplication (Class)

This interface is implemented by objects that can provide the basic services needed by a ChartII service application. These services include providing access to basic CORBA objects that are needed by service applications, such as the ORB, POA, Trader, and Event Service.

5.15.1.1.10ServiceApplicationModule (Class)

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

5.15.1.1.11UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.15.2 Sequence Diagrams

5.15.2.1 RoadwayLocation:ProvideCountyData (Sequence Diagram)

This sequence diagram describes how the system will get the data from the GIS mapping database.

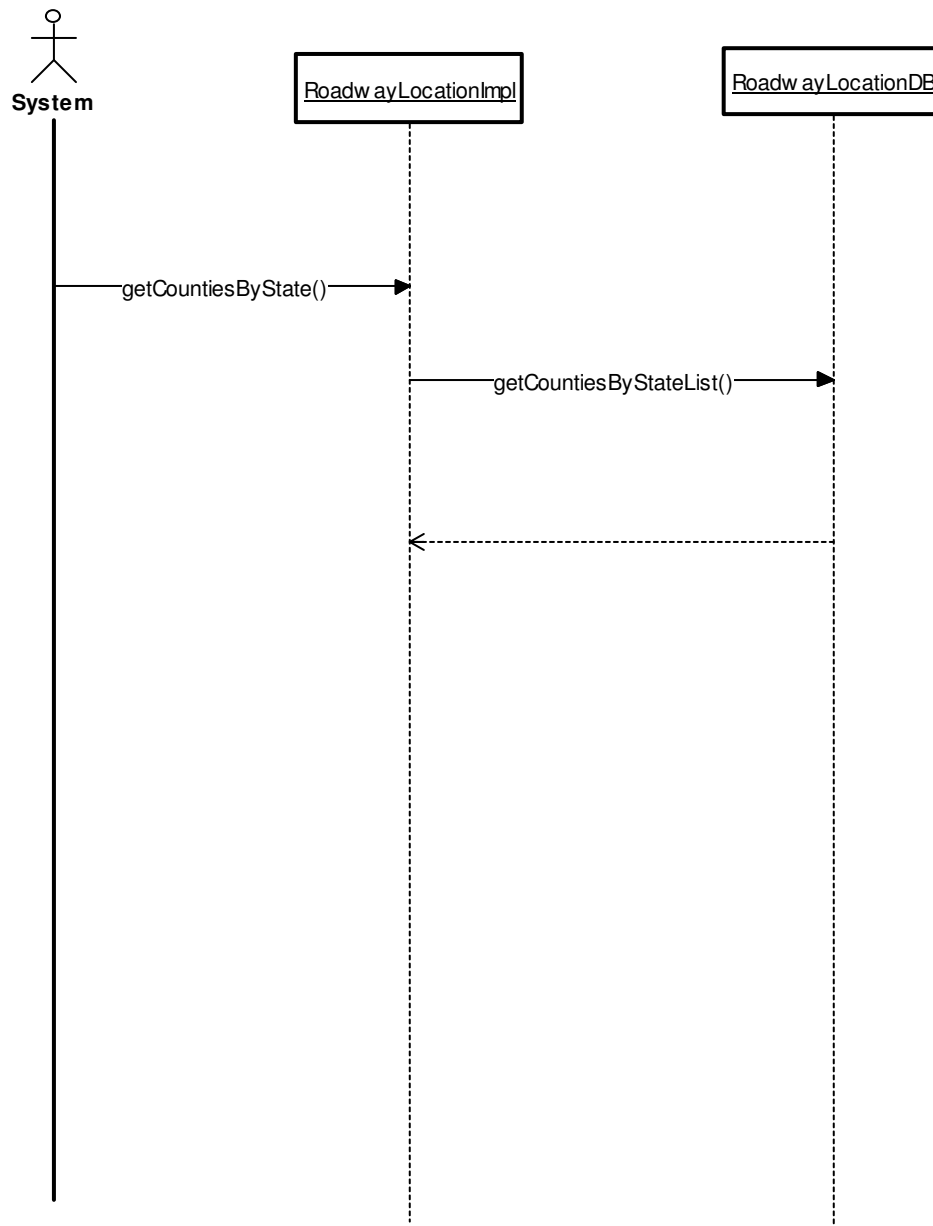


Figure 5-213. RoadwayLocation:ProvideCountyData (Sequence Diagram)

5.15.2.2 RoadwayLocationModule:Initialize (Sequence Diagram)

This sequence diagram shows the initialization of the RoadwayLocationModule on Service startup.

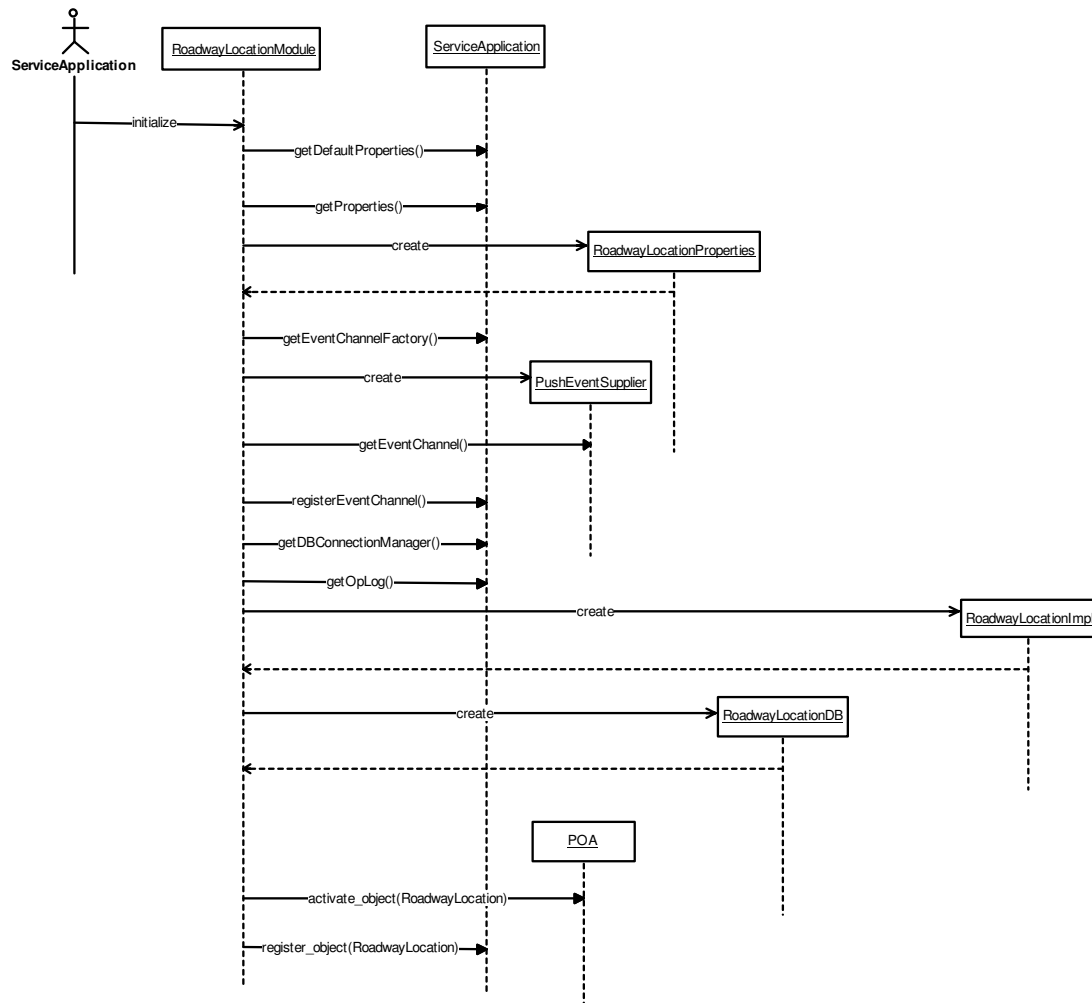


Figure 5-214. RoadwayLocationModule:Initialize (Sequence Diagram)

5.15.2.3 RoadwayLocationModule:Shutdown (Sequence Diagram)

This sequence diagram shows the processing when the ServiceApplication which contains the RoadwayLocationModule is shut down. The RoadwayLocationModule disconnects the RoadwayLocationImpl from the ORB and then tells it to shut down.

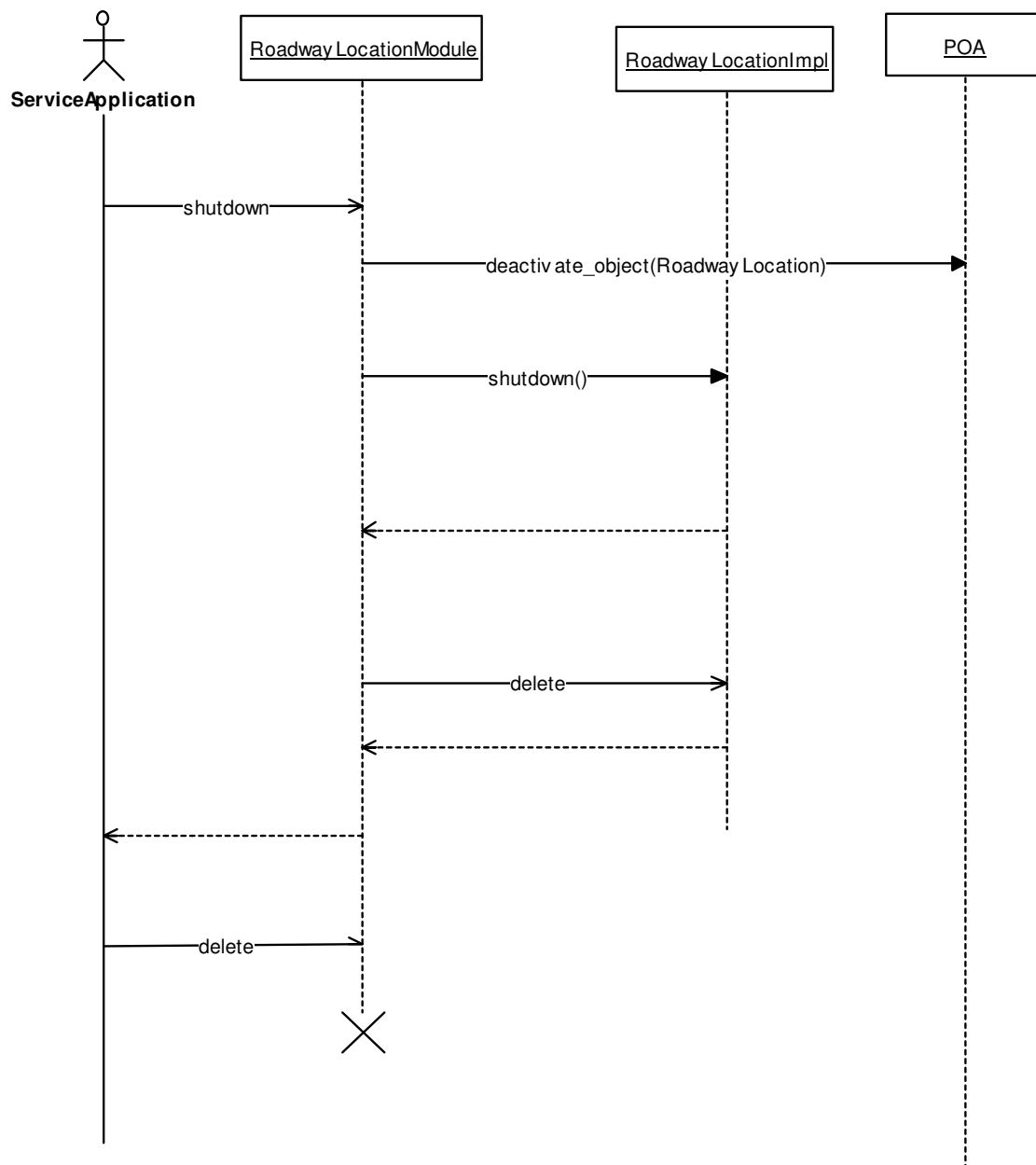


Figure 5-215. RoadwayLocationModule:Shutdown (Sequence Diagram)

5.16 SHAZAMControlModulePkg

5.16.1 Classes

5.16.1.1 SHAZAMControl (Class Diagram)

The SHAZAMControlModule serves a SHAZAMFactory object and SHAZAM objects. The class diagram below shows the classes used to implement these system interfaces.

reboot of a user's machine, because during a normal Chart II logout attempt, the logout is prohibited by Chart II system if the the user is the last user on his/her operations center and that operations center is controlling a maintenance mode sign. However, because anomalies happen, this task runs periodically to look for abandoned SHAZAM devices. This class implements the `java.util.TimerTask` interface, and as such it contains one method, `run()`, which is invoked by Java timer object on a regularly scheduled basis. This class contains a reference to the `SHAZAMFactoryImpl`, which is called upon to actually check the SHAZAM objects and controlling operations centers of each SHAZAM every time this task is called.

5.16.1.1.2 CommandQueue (Class)

The `CommandQueue` class provides a queue for `QueueableCommand` objects. The `CommandQueue` has a thread that it uses to process each `QueueableCommand` in a first in first out order. As each command object is pulled off the queue by the `CommandQueue`'s thread, the command object's `execute` method is called, at which time the command performs its intended task.

5.16.1.1.3 CommEnabled (Class)

The `CommEnabled` interface is implemented by objects that can be taken offline, put online, or put in maintenance mode through a standard interface. These states typically apply only to field devices. When a device is taken offline, it is no longer available for use through the system and automated polling (if any) is halted. When put online, a device is again available for use by `TrafficEvents` within the system and automated polling is enabled (if applicable). When put in maintenance mode a device is offline (i.e., cannot be used by `TrafficEvents`), and maintenance commands appropriate for the particular type of device are allowed to help in troubleshooting.

5.16.1.1.4 GeoLocatable (Class)

This interface is implemented by objects that can provide location information to their users.

5.16.1.1.5 HAR (Class)

This class is used to represent a Highway Advisory Radio (HAR) device. A HAR is used to broadcast traffic related information over a localized radio transmitter, making the information available to the traveler. This interface contains methods for getting and setting configuration, getting status, changing communications modes of a HAR, and manipulating and monitoring the HAR in maintenance and online modes.

5.16.1.1.6 HARMessageNotifier (Class)

The `HARMessageNotifier` class specifies an interface to be implemented by devices that can be used to notify the traveler to tune in to a radio station to hear a traffic message being broadcast by a HAR. A `HARMessageNotifier` is directional and allows users of the device to better determine if activation of the device is warranted for the message being broadcast

by the HAR. This interface can be implemented by SHAZAM devices and by DMS devices which are allowed to provide a SHAZAM-like message.

5.16.1.1.7 java.util.Timer (Class)

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

5.16.1.1.8 java.util.TimerTask (Class)

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

5.16.1.1.9 PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.16.1.1.10 QueueableCommand (Class)

A QueueableCommand is an interface used to represent a command that can be placed on a CommandQueue for asynchronous execution. Derived classes implement the execute method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a CommandQueue. The CommandQueue driver calls the execute method to execute a command in the queue and a call to the interrupted method is made when a CommandQueue is shut down.

5.16.1.1.11 RefreshSHAZAMTimerTask (Class)

The RefreshSHAZAMTimerTask class is responsible for refreshing all of the SHAZAM devices. This class implements the java.util.TimerTask interface, and as such it contains one method, run(), which is invoked by Java timer object on a regularly scheduled basis. This class contains a reference to the SHAZAMFactoryImpl, which is called upon to request each SHAZAM to refresh itself (command the device to its last known status) if its refresh interval has expired, each time this task is called.

5.16.1.1.12ServiceApplication (Class)

This interface is implemented by objects that can provide the basic services needed by a ChartII service application. These services include providing access to basic CORBA objects that are needed by service applications, such as the ORB, POA, Trader, and Event Service.

5.16.1.1.13ServiceApplicationModule (Class)

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

5.16.1.1.14SharedResource (Class)

The SharedResource interface is implemented by any object that may have an operations center responsible for the disposition of the resource while the resource is in use.

5.16.1.1.15SharedResourceManager (Class)

The SharedResourceManager interface is implemented by classes that manage shared resources. Implementing classes must be able to provide a list of all shared resources under their management. Implementing classes must also be able to tell others if there are any resources under its management that are controlled by a given operations center. The shared resource manager is also responsible for periodically monitoring its shared resources to detect if the operations center controlling a resource doesn't have at least one user logged into the system. When this condition is detected, the shared resource manager must push an event on the ResourceManagement event channel to notify others of this condition.

5.16.1.1.16SHAZAM (Class)

This interface class is used to identify the SHAZAM-specific methods which can be used to interface with a SHAZAM field device. It specifies methods for activating and deactivating the SHAZAM in maintenance mode, refreshing the SHAZAM (commanding the device to its last known status), changing the configuration of the SHAZAM, and removing the SHAZAM. This interface is implemented by a SHAZAMImpl class, which uses a helper ProtocolHdlr class to perform the model specific protocol for device command and control.

5.16.1.1.17SHAZAMActivateCmd (Class)

This class contains data needed to activate a SHAZAM asynchronously via the CommandQueue. A flag is used to determine if the activation is being performed directly on the device while it is in maintenance mode or if the activation is being processed as an extension of setting a HAR message in response to a traffic event.

5.16.1.1.18SHAZAMConfiguration (Class)

This class contains data that specifies the configuration of a SHAZAM device. It is used to

communicate configuration information to/from the database, and to/from the GUI clients. The GUI sends a SHAZAMConfiguration when creating a SHAZAM or modifying the configuration of an existing SHAZAM. Device Location member has been modified for R3B3. Now it contains a detailed location information.

5.16.1.1.19SHAZAMControlIDB (Class)

This class provides access to database functionality needed to support the SHAZAM and SHAZAMFactory classes. This class provides a high level interface to allow for persistence and depersistance of SHAZAM and SHAZAMFactory objects.

5.16.1.1.20SHAZAMControlModule (Class)

This class is a service module that provides control of SHAZAM devices. Upon initialization the module initializes a SHAZAMFactory which contains SHAZAM objects that have been previously added to the system. These objects are accessed via the CORBA ORB and manipulated directly from client applications. The module also creates support objects that are used by the SHAZAM (and SHAZAMFactory) objects to perform their processing, such as a database connection, event channels, and a periodic timer used to allow the objects to perform timer based processing.

5.16.1.1.21SHAZAMControlModuleProperties (Class)

This class is used to provide access to properties used by the SHAZAM Control Module. This class wraps properties that are passed to it upon construction. It adds its own defaults and provides methods to extract properties specific to the SHAZAM Control Module.

5.16.1.1.22SHAZAMDeactivateCmd (Class)

This class contains data needed to deactivate a SHAZAM asynchronously via the CommandQueue. A flag is used to determine if the deactivation is being performed directly on the device while it is in maintenance mode or if the deactivation is being processed as an extension of setting a HAR message in response to a traffic event.

5.16.1.1.23SHAZAMFactory (Class)

The SHAZAMFactory class specifies the interface to be used to create SHAZAM objects within the Chart II system. It also provides a method to get a list of SHAZAM devices currently in the system.

5.16.1.1.24SHAZAMFactoryImpl (Class)

This class provides the ability to add new SHAZAM objects to the system. When SHAZAMs are added, they are persisted to the database so this object can depersist them upon startup. This class also provides a removeSHAZAM method that allows a SHAZAM to remove itself from the system when directed. This class is also responsible for performing the checks requested by the timer tasks: to refresh the SHAZAM devices and to look for SHAZAM devices with no one logged in at the controlling operations center.

5.16.1.1.25SHAZAMImpl (Class)

The SHAZAMImpl class provides an implementation of the SHAZAM interface, and by extension the SharedResource, HARMMessageNotifier, CommEnabled, GeoLocatable, and UniquelyIdentifiable interfaces as specified by the IDL.

This class contains a CommandQueue object that is used to sequentially execute long running operations (field communications to the device) in a thread separate from the CORBA request threads, thus allowing quick initial responses.

Also contained in this class are SHAZAMConfiguration and SHAZAMStatus objects (used to store the configuration and status of the sign), a lastRefreshTime value used for refreshing (commanding the device to its last known status), and a list of TrafficEvent objects that are currently active on the SHAZAM.

The SHAZAMImpl contains *Impl methods that map to methods specified in the IDL, including requests to activate and deactivate the SHAZAM, put the SHAZAM online, put the SHAZAM offline, put the SHAZAM in maintenance mode, or to change (set) the configuration of the SHAZAM. All of these requests require (or potentially require) field communications to the device, so each request is stored in a specific subclass of QueueableCommand and added to the CommandQueue. The queueable command objects simply call the appropriate SHAZAMImpl method as the command is executed by the CommandQueue in its thread of execution.

The SHAZAMImpl also contains methods called by the SHAZAMFactory to support the timer tasks of the SHAZAM Service: to refresh the SHAZAM devices and to look for maintenance mode SHAZAM devices with no one logged in at the controlling operations center.

5.16.1.1.26SHAZAMPutInMaintModeCmd (Class)

This command contains data needed to put a SHAZAM device in maintenance mode (from either offline or online mode) asynchronously via the CommandQueue. When executed this class calls back into the SHAZAMImpl object to execute the putInMaintenanceModeImpl method.

5.16.1.1.27SHAZAMPutOnlineCmd (Class)

This command contains data needed to put a SHAZAM device online (from maintenance or offline mode) asynchronously via the CommandQueue. When executed this class calls back into the SHAZAMImpl object to execute its putOnLineImpl method.

5.16.1.1.28SHAZAMRefreshCmd (Class)

This class is a command object used to invoke the SHAZAM refresh processing (commanding the device to its last known status) asynchronously from the command queue. When executed, this class calls back into the SHAZAMImpl object to execute the refreshImpl method.

5.16.1.1.29SHAZAMSetConfigurationCmd (Class)

This command contains data needed to set the SHAZAM configuration asynchronously via the CommandQueue. When executed, this class calls back into the SHAZAMImpl object to execute its setConfigurationImpl method. The SHAZAM device model currently in use does not contain any configuration settings, however this command is still processed asynchronously for consistency.

5.16.1.1.30SHAZAMStateAction (Class)

The SHAZAMStateAction class enumerates the types of actions (commands) that set the state of a SHAZAM: ACTIVATE or DEACTIVATE.

5.16.1.1.31SHAZAMStatus (Class)

This class contains the current status of a SHAZAM device. This class is used to store status within the SHAZAM object, and is also used to communicate configuration information to/from the database, and to the GUI clients (one-way).

5.16.1.1.32SHAZAMTakeOfflineCmd (Class)

This command contains data needed to take a SHAZAM device offline (from online or maintenance mode) asynchronously via the CommandQueue. When executed, this class calls back into the SHAZAMImpl object to execute its takeOfflineImpl method.

5.16.1.1.33TokenManipulator (Class)

This class contains all functionality required for user rights in the system. It is the only code in the system which knows how to create, modify and check a user's functional rights. It encapsulates the contents of an octet sequence which will be passed to every secure method. Secure methods should call the checkAccess method to validate the user. Client processes should use the check access method to verify access and optimize to reduce the size of the sequence to only those rights which are necessary to invoke the secure method. The token contains the following information. Token version, Token ID, Token Time Stamp, Username, Op Center ID, Op Center IOR, functional rights

5.16.1.1.34TrafficEvent (Class)

Objects of this type represent traffic events that require action from system operators.

5.16.1.1.35UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.16.1.1.36VikingRC2AProtocolHdlr (Class)

This protocol handler contains the protocol used to communicate with a Viking RC2A SHAZAM device.

5.16.2 SequenceDiagrams

5.16.2.1 SHAZAMControlModule:setConfiguration (Sequence Diagram)

A user with appropriate functional rights can set the configuration of a SHAZAM if it is in maintenance mode. The Rc2aSHAZAM itself does not have any configurable settings, so no field communications are necessary. Although this command does not currently require field communications, the asynchronous command pattern is used for consistency with other device commands and also to allow the code to easily adapt to a device type that supports configurable settings. When the command is executed, setConfigurationImpl stores configuration in memory. If it is communication parameters that have changed, a new VoicePortLocator is created. The new configuration is persisted to the database and an event is pushed onto the status event channel to notify others of the changes.

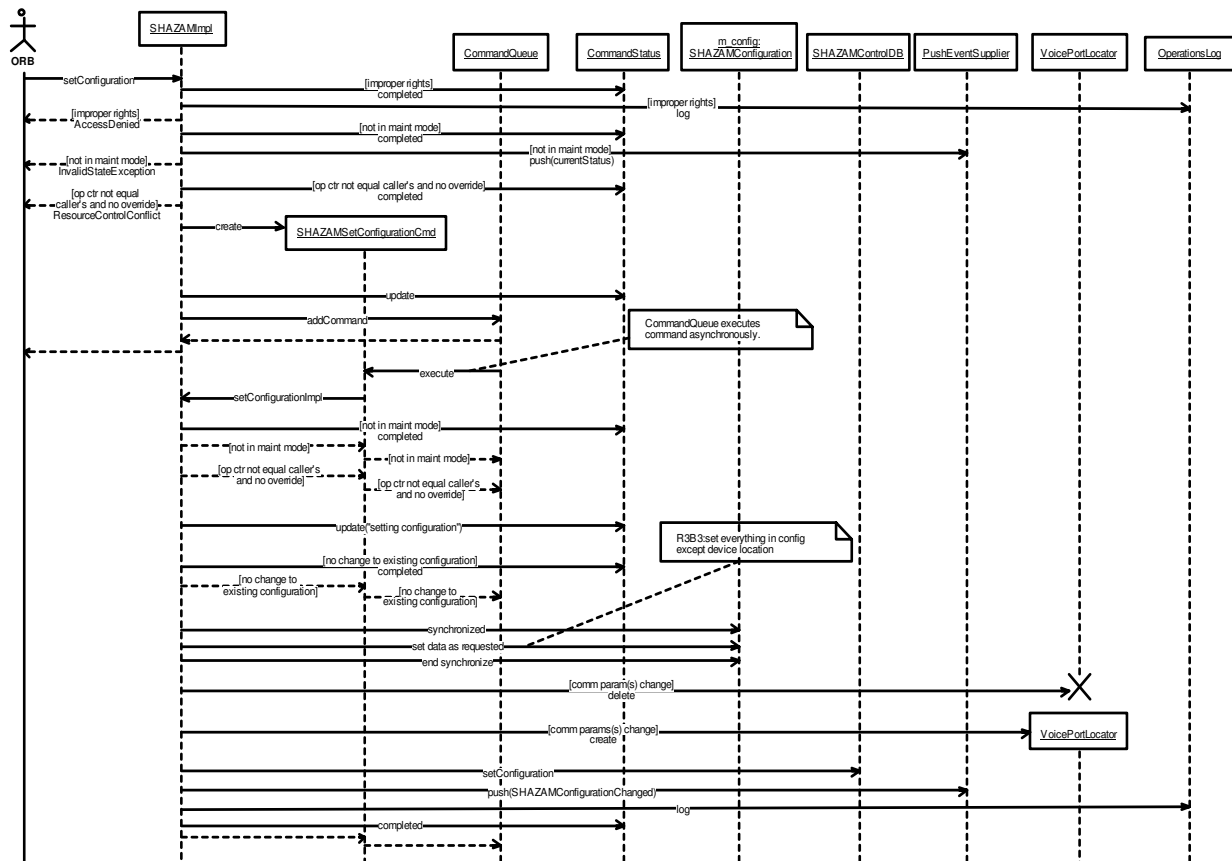


Figure 5-217. SHAZAMControlModule:setConfiguration (Sequence Diagram)

5.17 SHAZAMManagementPkg

5.17.1 Classes

5.17.1.1 SHAZAMUtility (Class Diagram)

This diagram shows SHAZAM related classes that are shared between the server and the GUI.

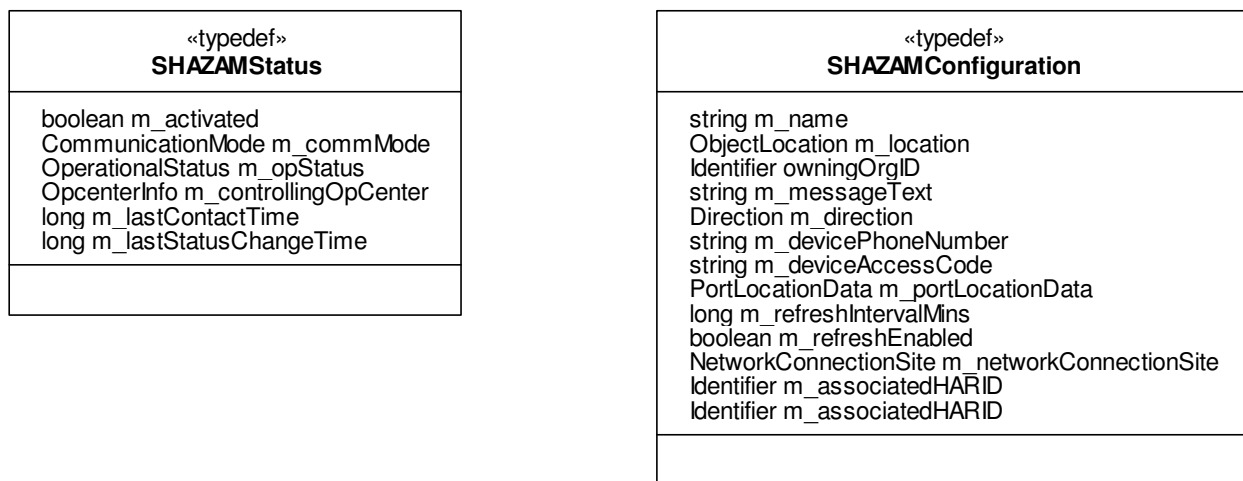


Figure 5-218. SHAZAMUtility (Class Diagram)

5.17.1.1.1 SHAZAMConfiguration (Class)

This class contains data that specifies the configuration of a SHAZAM device. It is used to communicate configuration information to/from the database, and to/from the GUI clients. The GUI sends a **SHAZAMConfiguration** when creating a SHAZAM or modifying the configuration of an existing SHAZAM. Device Location member has been modified for R3B3. Now it contains a detailed location information.

5.17.1.1.2 SHAZAMStatus (Class)

This class contains the current status of a SHAZAM device. This class is used to store status within the SHAZAM object, and is also used to communicate configuration information to/from the database, and to the GUI clients (one-way).

5.18 System Interfaces

5.18.1 Classes

5.18.1.1 AlertManagement (Class Diagram)

This class diagram shows the system interfaces that make the AlertManagement capability of CHART2 system.

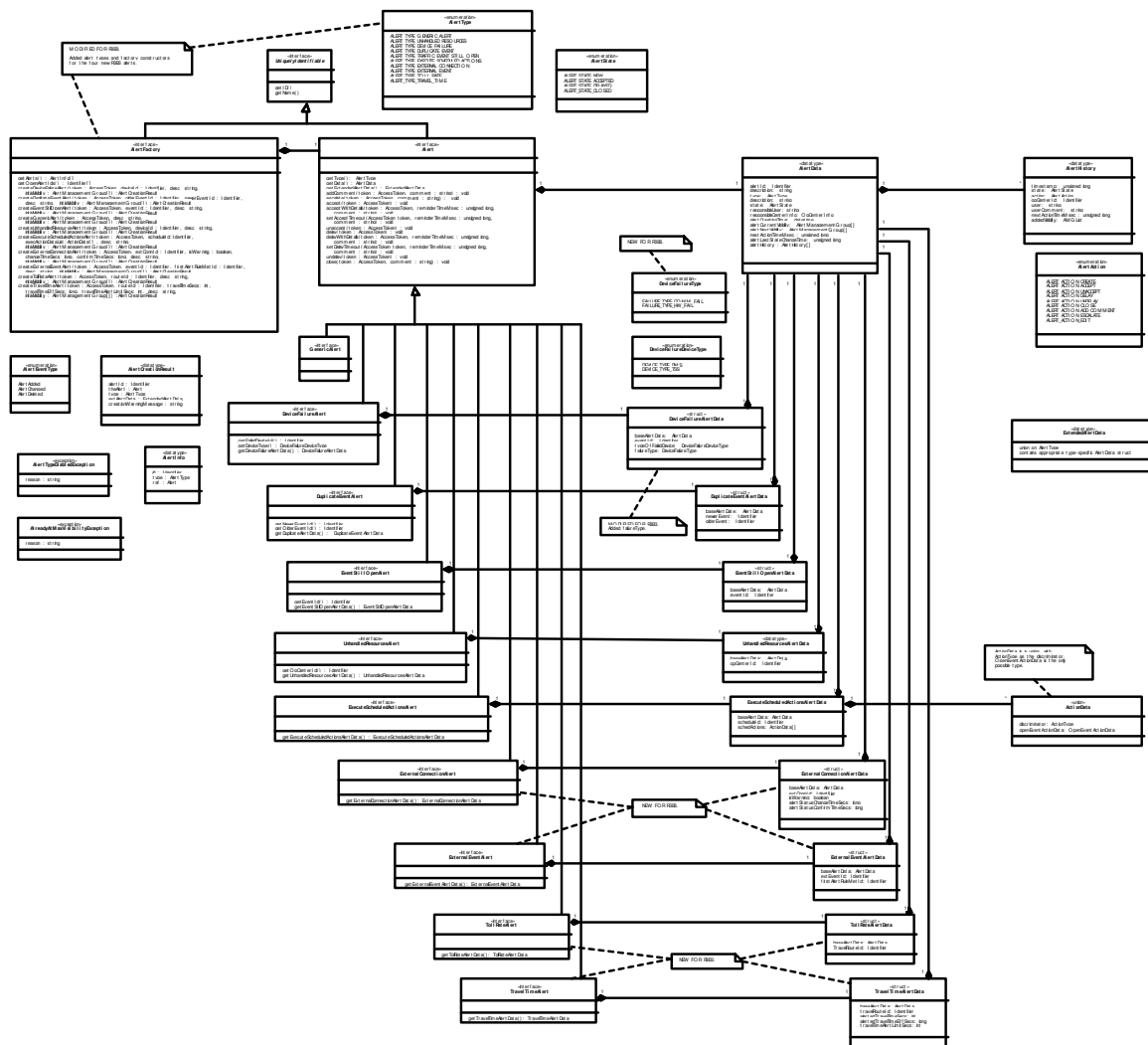


Figure 5-219. AlertManagement (Class Diagram)

5.18.1.1.1 ActionData (Class)

This IDL union holds the data used to describe a schedule action. It has been designed as a union discriminated by the enumeration ActionType to support schedule actions to be determined in future releases of CHART. Currently the only supported variant is the OpenEventAction.

5.18.1.1.2 Alert (Class)

This is a CORBA interface that provides access to information pertaining to an Alert and provides operations used to manage an alert.

5.18.1.1.3 AlertAction (Class)

This IDL enumeration defines the actions that can be done to an Alert.

5.18.1.1.4 AlertCreationResult (Class)

This IDL struct represents the data that will be returned as a result of an alert creation using the AlertFactory calls. It includes: alert id, alert CORBA reference, alert type, extended alert data, and a warning string used to describe non-fatal conditions when creating the alert.

5.18.1.1.5 AlertData (Class)

This is a CORBA struct, defined in IDL, that contains the data that applies to all alert types.

5.18.1.1.6 AlertEventType (Class)

This IDL enumeration defines the types of CORBA Events supported in the AlertModule. Its primary use is as a discriminator value used when handling AlertEvents. These can either be Alert Added, Changed, or Deleted.

5.18.1.1.7 AlertFactory (Class)

This IDL interface contains the operations available for an Alert Factory. The AlertFactory is responsible for creating alerts and storing alert information on the alerts that it created.

5.18.1.1.8 AlertHistory (Class)

This IDL struct contains information used to describe an action being done to an alert. A collection of these structs represents the history of the alert from beginning to end.

5.18.1.1.9 AlertInfo (Class)

This IDL struct contains information about an Alert in the system. Its primary use is to be returned as part of a list of AlertInfo objects in response to an AlertFactory's getAlerts() call.

5.18.1.1.10AlertState (Class)

AlertState is an IDL enumeration of the four defined states for an Alert.

5.18.1.1.11AlertType (Class)

AlertType is an IDL enumeration of the five Alert types.

5.18.1.1.12AlertTypeDisabledException (Class)

This exception is thrown by the AlertFactory create operations if the alert type being created is disabled within the system. (Server-side clients can ignore this alert; GUI-side clients may wish to display this to the user.)

5.18.1.1.13AlreadyAtMaxVisibilityException (Class)

This exception is thrown by the Alert escalate() operation if the alert is already at maximum visibility (no additional AMGs are configured in the backup set(s) of the AMG(s) in the current visibility list). Clients may wish to try escalation after receipt of this exception (or at any time the nextVisibility array is empty), in case an administrator may have modified the backup set of AMGs in the meanwhile.

5.18.1.1.14DeviceFailureAlert (Class)

This IDL interface contains operations specific to a Device Failure alert. This interface is implemented by classes representing DeviceFailureAlerts in the Chart2 System.

5.18.1.1.15DeviceFailureAlertData (Class)

This is a CORBA struct, defined in IDL, that contains base alert data plus data specific to a DeviceFailureAlert. Specific to this alert is the traffic event id of the failed device event causing the alert. Also included is information on the device failure type.

5.18.1.1.16DeviceFailureDeviceType (Class)

The DeviceFailureDeviceType is an enumeration of the possible device failure types supported in a device failure alert.

5.18.1.1.17DeviceFailureType (Class)

This enumeration lists the possible types of device failures which can be communicated by a device failure alert.

5.18.1.1.18DuplicateEventAlert (Class)

This IDL interface contains operations specific to a Duplicate Event alert. This interface is implemented by classes representing DuplicateEventAlertsDevice in the Chart2 System.

5.18.1.1.19DuplicateEventAlertData (Class)

This is a CORBA struct, defined in IDL, that contains base alert data plus data specific to a DuplicateEventAlert. Specific to this alert are the event ids of the two probable duplicate traffic events.

5.18.1.1.20EventStillOpenAlert (Class)

This IDL interface contains operations specific to a Event Still Open alert. This interface is implemented by classes representing EventStillOpenAlerts in the Chart2 System.

5.18.1.1.21EventStillOpenAlertData (Class)

This is a CORBA struct, defined in IDL, that contain the base alert data plus data specific to an EventStillOpenAlert. Specific to this alert is the id of the traffic event that is still open.

5.18.1.1.22ExecuteScheduledActionsAlert (Class)

This IDL interface contains operations specific to aExecute Scheduled Actions alert. This interface is implemented by classes representing ExecuteScheduledActionsAlert in the Chart2 System.

5.18.1.1.23ExecuteScheduledActionsAlertData (Class)

This is a CORBA struct, defined in IDL, that contains the base alert data plus data specific to an ExecuteScheduledActionsAlert.

5.18.1.1.24ExtendedAlertData (Class)

ExtendedAlertData is a union of the four type specific alert datatypes: DeviceFailureAlertData, DuplicateEventAlertData, EventStillOpenAlertData, and UnhandledResourceAlertData. Note that the GenericAlert does not include any type specific data. The AlertType enumeration is used as the discriminator over the data in this union.

5.18.1.1.25ExternalConnectionAlert (Class)

This IDL interface contains operations specific to an External Connection Alert, which indicates trouble with a connection between CHART and an external system.

5.18.1.1.26ExternalConnectionAlertData (Class)

This IDL structure contains data specific to an External Connection Alert, e.g., the ID of the interface which is having trouble and a flag indicating whether the connection is in failure or warning status, the timestamp it transitioned. (The GUI displays additional data which is best acquired from the GUI's object cache.) (Text in the base AlertData structure provides a textual description and alert management data.)

5.18.1.1.27ExternalEventAlert (Class)

This IDL interface contains operations specific to an External Event Alert, which indicates an event has arrived from an external system which satisfies criteria a CHART administrator has defined to flag an external event as significant enough to warrant this alert.

5.18.1.1.28ExternalEventAlertData (Class)

This IDL structure contains data specific to an External Event Alert, e.g., the ID of the event and the ID of the first rule found that requested an alert be sent. (Text in the base AlertData structure provides a textual description and alert management data.)

5.18.1.1.29GenericAlert (Class)

This IDL interface contains operations specific to a Generic alert. This interface is implemented by classes representing GenericAlerts in the Chart2 System.

5.18.1.1.30TollRateAlert (Class)

This IDL interface contains operations specific to an Toll Rate Alert, which indicates a travel route which had a currently active toll rate no longer does in a more recently received toll rate update document from a toll rate provider. (This alert is not sent if a toll rate expires due to an absence of any current toll rate document -- such an event would have triggered one external connection alert and does not need to also trigger a multitude of individual toll rate alerts as well.)

5.18.1.1.31TollRateAlertData (Class)

This IDL structure contain data specific to a Toll Rate Alert, e.g., the travel route which no longer has data for its toll rate. (Text in the base AlertData structure provides a textual description and alert management data.)

5.18.1.1.32TravelTimeAlert (Class)

This IDL interface contains operations specific to an Travel Time Alert, which indicates the travel time associated with a travel route is high enough to warrant this alert.

5.18.1.1.33TravelTimeAlertData (Class)

This IDL structure contains data specific to a Travel Time Alert, e.g., the travel time limit and the travel time which exceeded the limit. (Text in the base AlertData structure provides a textual description and alert management data.)

5.18.1.1.34UnhandledResourcesAlert (Class)

This IDL interface contains operations specific to a Unhandled Resources alert. This interface is implemented by classes representing UnhandledResourceAlerts in the Chart2 System.

5.18.1.1.35 UnhandledResourcesAlertData (Class)

This is a CORBA struct, defined in IDL, that contains the base alert data plus data specific to an UnhandledResourcesAlert.

5.18.1.1.36 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.18.1.2 Common (Class Diagram)

This class diagram shows classes used by multiple modules.

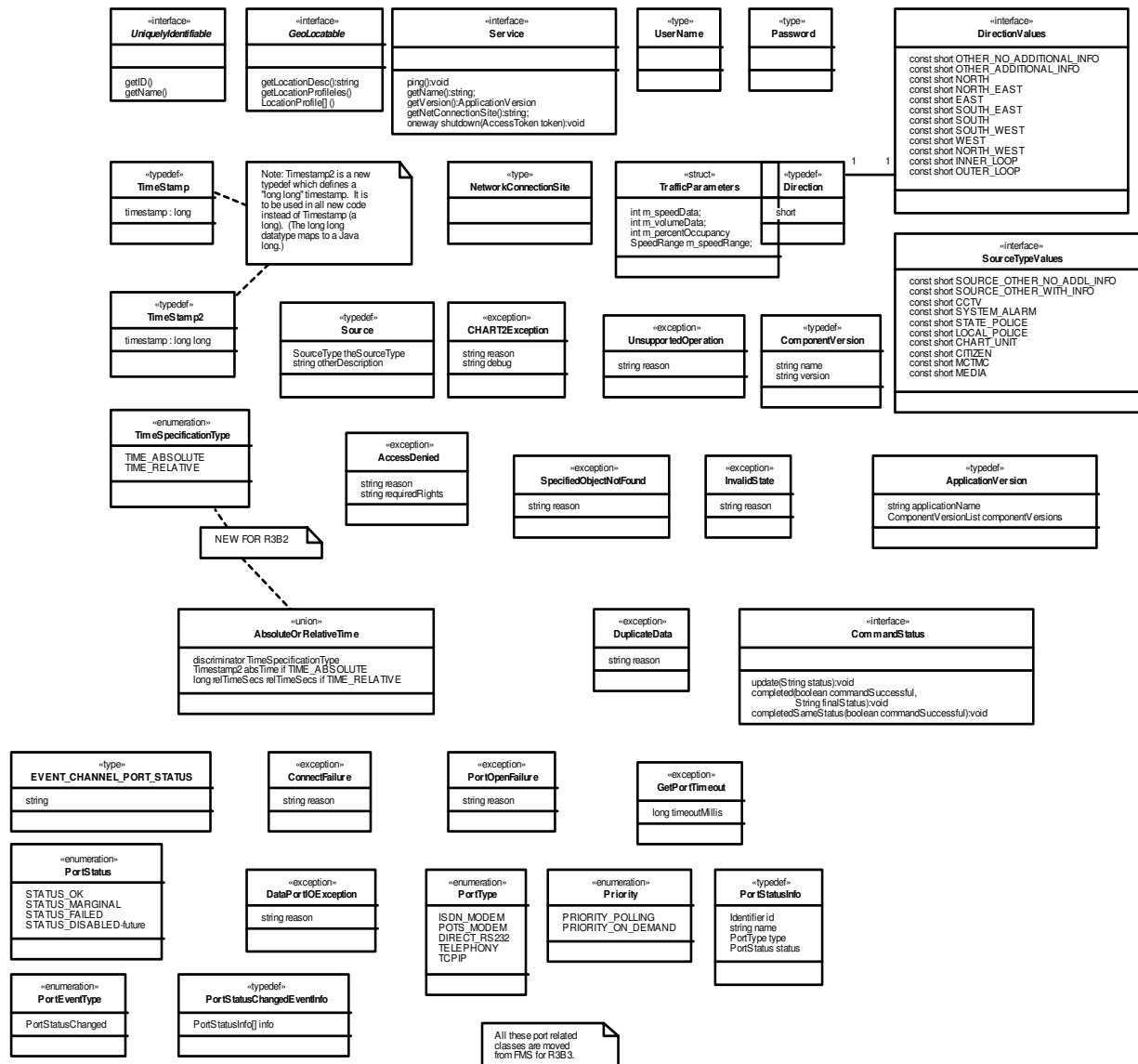


Figure 5-220. Common (Class Diagram)

5.18.1.2.1 AbsoluteOrRelativeTime (Class)

This union stores a time, in either absolute or relative terms.

5.18.1.2.2 AccessDenied (Class)

This class represents an access denied, or "no rights" failure.

5.18.1.2.3 ApplicationVersion (Class)

This structure contains the name of the application and information about the versions of its components.

5.18.1.2.4 CHART2Exception (Class)

Generic exception class for the CHART2 system. This class can be used for throwing very generic exceptions which require no special processing by the client. It supports a reason string which may be shown to any user and a debug string which will contain detailed information useful in determining the cause of the problem.

5.18.1.2.5 CommandStatus (Class)

The CommandStatus CORBA interface is used to allow a calling process to be notified of the progress of a long-running asynchronous operation. This is normally used when field communications are involved to complete a method call. The most common use is to allow a GUI to show the user the progress of an operation. It can also be used and watched by a server process when it needs to call on another server process to complete an operation. The long running operation typically calls back to the CommandStatus object periodically as the command is being executed, to provide in-progress status information, and it always makes a final call to the CommandStatus when the operation has completed. The final call to the CommandStatus from the long running operation indicates the success or failure of the command.

5.18.1.2.6 ComponentVersion (Class)

This structure contains the name and version number of the software component.

5.18.1.2.7 ConnectFailure (Class)

This exception is a catch-all for exceptions that do not fit in a more specific exception that can be thrown during a connection attempt.

5.18.1.2.8 DataPortIOException (Class)

This exception is used to indicate an Input/Output error has occurred.

5.18.1.2.9 Direction (Class)

This type defines a short value that is used to indicate a direction of travel as defined in DirectionValues.

5.18.1.2.10 DirectionValues (Class)

This interface contains constants for directions as defined in the TMDD.

5.18.1.2.11DuplicateData (Class)

This exception is thrown when an object is to be added to the system, but the system already contains an object with equivalent data.

5.18.1.2.12EVENT_CHANNEL_PORT_STATUS (Class)

This is a static string that contains the name of the event channel used to push events relating to the change in Port status. The following PortEventTypes are pushed on EVENT_CHANNEL_PORT_STATUS channel: PortStatusChanged

5.18.1.2.13GeoLocatable (Class)

This interface is implemented by objects that can provide location information to their users.

5.18.1.2.14GetPortTimeout (Class)

This class is an exception that is thrown by a PortManager when a request to acquire a port of a given type cannot be fulfilled within the timeout specified.

5.18.1.2.15InvalidState (Class)

This exception is thrown when an operation is attempted on an object that is not in a valid state to perform the operation.

5.18.1.2.16NetworkConnectionSite (Class)

The NetworkConnectionSite class contains a string that is used to specify where a service is running. This field is useful for administrators in debugging problems should an object become "software comm failed".. It is included in the Chart2DMSSStatus.

5.18.1.2.17Password (Class)

Typedef used to define the type of a Password.

5.18.1.2.18PortEventType (Class)

This enum defines the types of CORBA events that are pushed on a Field Communications event channel.

5.18.1.2.19PortOpenFailure (Class)

This exception is thrown if there is an error opening the port while attempting a connection. This exception would most likely only occur if there is another application accessing the physical com port, which would be true if debugging activities were being done on a port while the FieldCommunications service is still running.

5.18.1.2.20PortStatus (Class)

This enumeration specifies the values used to represent a Port's status. OK signifies the port is working properly. MARGINAL signifies errors have been experienced during recent use of the port. FAILED indicates the port is not working at all.

5.18.1.2.21PortStatusChangedEventInfo (Class)

This class contains data that is pushed on a Field Communications event channel with a PortStatusChanged event.

5.18.1.2.22PortStatusInfo (Class)

This class contains the data of status of a particular port.

5.18.1.2.23PortType (Class)

This enumeration defines the types of ports that may be requested from a PortManager.

5.18.1.2.24Priority (Class)

This enumeration specifies the priority levels used when requesting a port from a PortManager. ON_DEMAND is given higher priority than POLLING.

5.18.1.2.25Service (Class)

This interface is implemented by all services in the system that allow themselves to be shutdown externally. All implementing classes provide a means to be cleanly shutdown and can be pinged to detect if they are alive.

5.18.1.2.26Source (Class)

This structure contains information about the source of the data being added to the system.

5.18.1.2.27SourceTypeValues (Class)

This enumeration contains the possible sources of information that can be used for adding CommLog entries and/or traffic event data.

5.18.1.2.28SpecifiedObjectNotFound (Class)

Exception used to indicate that an operation was attempted that involves a secondary object that cannot be found by the invoked object.

5.18.1.2.29TimeSpecificationType (Class)

This enumeration lists the types of times which can be stored in the AbsoluteOrRelativeTime union.

5.18.1.2.30TimeStamp (Class)

This typedef defines the type of TimeStamp fields.

5.18.1.2.31TimeStamp2 (Class)

This data type offers extended date range beyond the year 2038 limitation implicit in the TimeStamp data type.

5.18.1.2.32TrafficParameters (Class)

This struct contains traffic parameters that are sensed and reported by a Traffic Sensor System such as the RTMS.

m_speedData - The arithmetic mean of the speeds collected over a sample period in miles per hour in tenths. (thus 550 == 55.0 MPH) Valid values are 0 to 2550. A value of 65535 is used to indicate a missing or invalid value (such as when the volume for the sample period is zero).

m_volumeData - The count of vehicles for the sample period. Valid values 0 to 65535. A value of 65535 represents a missing value.

m_percentOccupancy - The percentage of occupancy of the roadway in tenths of a percent. (thus 1000 = 100.0 percent). Valid values are 0 to 1000. A value of 65535 represents a missing or invalid value.

5.18.1.2.33UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.18.1.2.34UnsupportedOperation (Class)

This exception is used to indicate that an operation is not supported by the object on which it is called.

5.18.1.2.35UserName (Class)

This typedef defines the type of UserName fields used in system interfaces.

5.18.1.3 Common2 (Class Diagram)

This class diagram shows classes used by many other modules within the CHART System. This diagram supplements the “Common” Class Diagram, showing additional classes which cannot fit on the original “Common” Class Diagram.

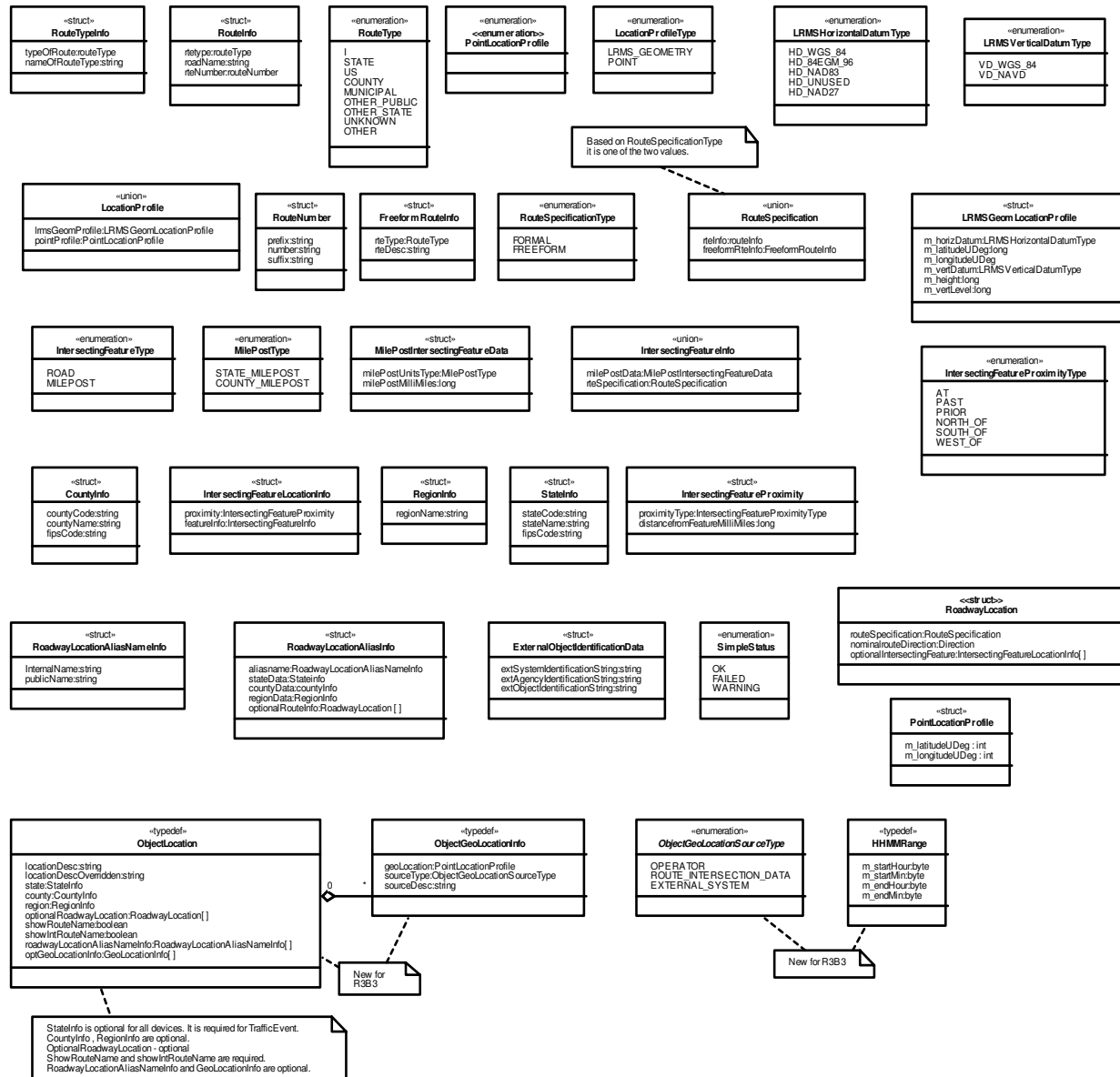


Figure 5-221. Common2 (Class Diagram)

5.18.1.3.1 <<enumeration>> PointLocationProfile

This structure contains the latitude and longitude of geographical location

5.18.1.3.2 <<struct>> RoadwayLocation (Class)

This structure has the information to define roadwaylocation of CHART objects like devices and TrafficEvents

5.18.1.3.3 CountyInfo (Class)

This structure contains information about a county.

5.18.1.3.4 ExternalObjectIdentificationData (Class)

This structure is used to hold data which identifies the external source of an external object which has been imported into CHART.

5.18.1.3.5 FreeformRouteInfo (Class)

Information specifying a route when only the route type and route description are known, such as may be the case when a route is entered by the user.

5.18.1.3.6 HHMMRange (Class)

This structure defines a time duration.

5.18.1.3.7 IntersectingFeatureInfo (Class)

This union provides auxiliary data for identifying an intersecting feature along a given roadway.

5.18.1.3.8 IntersectingFeatureLocationInfo (Class)

Location of a point along a roadway, using an intersecting feature to define the location.

5.18.1.3.9 IntersectingFeatureProximity (Class)

This structure defines a location of a point on a given roadway relative to an intersecting feature.

5.18.1.3.10IntersectingFeatureProximityType (Class)

This enumeration represents a direction relative to an intersecting feature on a roadway for defining a location on the roadway.

5.18.1.3.11IntersectingFeatureType (Class)

The type of intersecting feature which is used to define a point along a given roadway.

5.18.1.3.12LocationProfile (Class)

Data included in an LRMS geometry location profile.

5.18.1.3.13LocationProfileType (Class)

Defines all supported location profiles for GeoLocatable objects in the system.

5.18.1.3.14LRMSGeomLocationProfile (Class)

Data included in an LRMS geometry location profile.

5.18.1.3.15LRMSHorizontalDatumType (Class)

This enum lists the values that can be used for horizontal datum for the LRMS (Location Referencing Message Specification) Geometry profile using TMDD proscribed values for loc_ext_horizontal_datum. (Reference TMDD Vol II Annex June 2004).

5.18.1.3.16LRMSVerticalDatumType (Class)

This enum lists the values that can be used for vertical datum for the LRMS (Location Referencing Message Specification) Geometry profile using TMDD proscribed values for loc_ext_vertical_datum. (Reference TMDD Vol II Annex June 2004).

5.18.1.3.17MilePostIntersectingFeatureData (Class)

This structure defines a milepost location along a given roadway, with a milepost measurement in terms of the specified milepost type.

5.18.1.3.18MilePostType (Class)

This enumeration lists the type of milepost units.

5.18.1.3.19ObjectGeoLocationInfo (Class)

This structure defines the geographical location of a CHART object.

5.18.1.3.20ObjectGeoLocationSourceType (Class)

This structure defines the source of geolocaition information of an object(a CHART entity).

5.18.1.3.21ObjectLocation (Class)

This structure defines the location of CHART objects like devices and traffic events. StateInfo, CountyInfo, RegionInfo, RoadwayLocation, RoadwayLocationAliasNameInfo and GeoLocationInfo fields are optional.

5.18.1.3.22PointLocationProfile (Class)

This struct represents a geographical point defined as a latitude / longitude pair. The latitude and longitude are defined in microdegrees.

5.18.1.3.23RegionInfo (Class)

This structure contains information about a region.

5.18.1.3.24RoadwayLocationAliasInfo (Class)

This structure contains the aliases for locations. For example, an alias can describe the Fort McHenry Tunnel where the alias would be FMT.

5.18.1.3.25RoadwayLocationAliasNameInfo (Class)

This structure contains information on the two names of an alias for a roadway location.

5.18.1.3.26RouteInfo (Class)

Information for specifying a route when the components of the route number information (and optionally the route name) are known.

5.18.1.3.27RouteNumber (Class)

A route number, which may consist of an alphanumeric prefix, a number, and an alphanumeric suffix.

5.18.1.3.28RouteSpecification (Class)

This union specifies a route using either a formal definition or a freeform text definition.

5.18.1.3.29RouteSpecificationType (Class)

This enum indicates whether a route is specified using the formal definition (as is the case when the route number components are known), or whether the route was entered as freeform text.

5.18.1.3.30RouteType (Class)

This enumeration is used to specify the classification of a road (interstate, MD, etc.)

5.18.1.3.31RouteTypeInfo (Class)

This structure contains information about the classification type of a road.

5.18.1.3.32SimpleStatus (Class)

This enum defines simple status values.

5.18.1.3.33StateInfo (Class)

This structure contains information about a State.

5.18.1.4 DMSControl (Class Diagram)

This Class Diagram shows the CORBA system interface classes and methods used to manipulate DMS services within the CHART system. This diagram was modified for R3B3 to add support for traveler information messages, device location, TCP/IP communications, NTCIP font control, notifications, and more alerts.

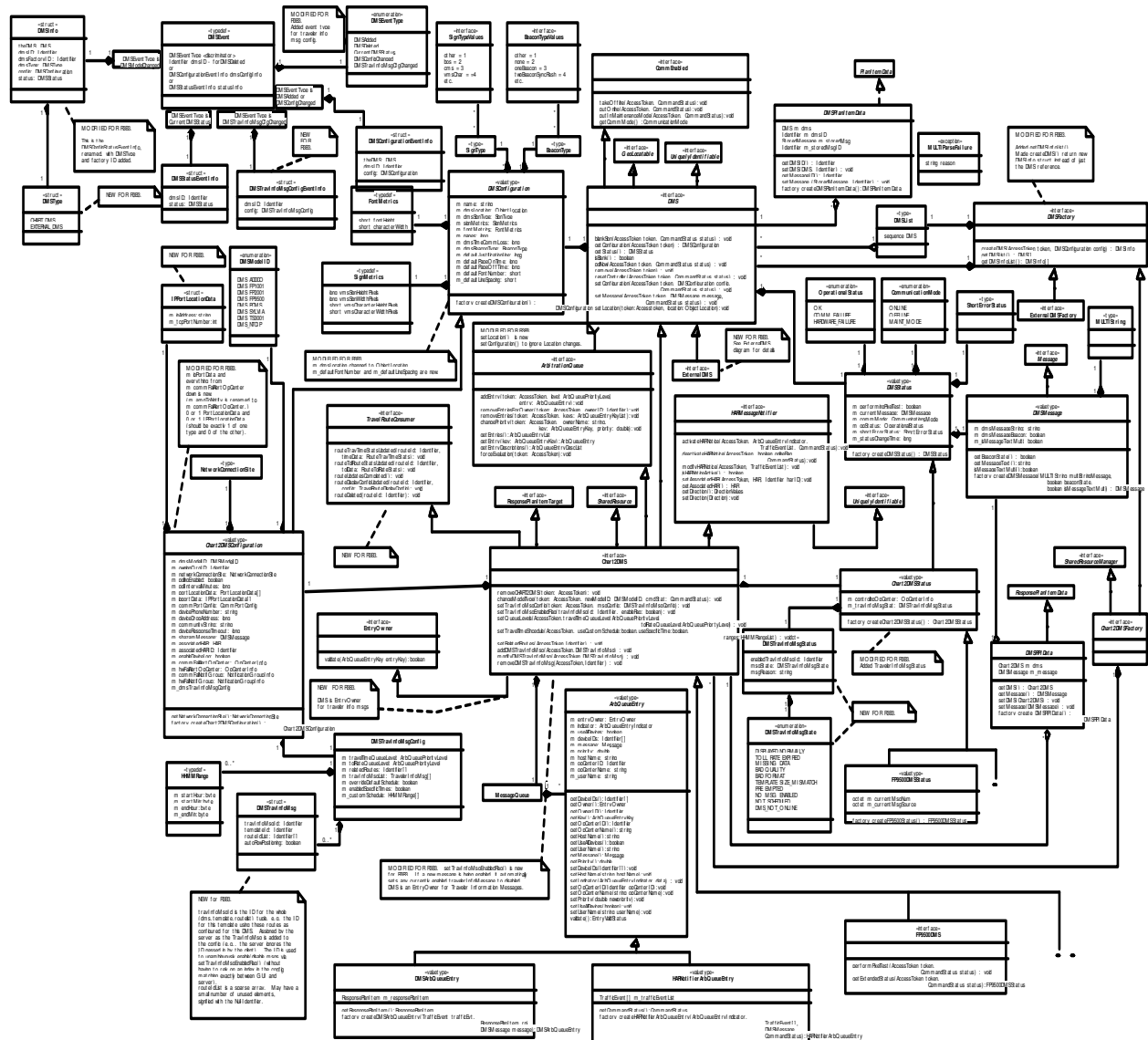


Figure 5-222. DMSCControl (Class Diagram)

5.18.1.4.1 ArbitrationQueue (Class)

An ArbitrationQueue is a queue that arbitrates the usage of a device. The evaluation of the queue determines which message(s) should be on the device, based upon the priority of the

queue entries. When entries are added to the queue, they are assigned a priority level based on the type of traffic event with which they are associated, and also upon the current contents of the queue. The priority of the queue entries can be modified after they are added to the queue. The queue is evaluated when the device is online and queue entries are added or removed, when an entry's priority is modified, or when the device is put online.

5.18.1.4.2 ArbQueueEntry (Class)

This class is used for an entry on the arbitration queue, for a single message, and for a single traffic event. (It is possible, in the case of HARNotifierArbQueueEntry objects, that certain ArbQueueEntries can be on behalf of multiple TrafficEvents. In such cases, one TrafficEvent among all those involved is picked to be the responsible TrafficEvent stored in m_indicator, the ArbQueueEntryIndicator for the entry.)

5.18.1.4.3 BeaconType (Class)

The BeaconType class defines the beacon type for a DMS. Its values are defined by the BeaconTypeValues class. It is a part of a DMSConfiguration object.

5.18.1.4.4 BeaconTypeValues (Class)

The BeaconTypeValues class enumerates the various beacon types used on DMS devices (number of beacons and whether and in what manner they flash).

5.18.1.4.5 Chart2DMS (Class)

The Chart2DMS class extends the DMS interface and defines a more detailed interface to be used in manipulating the CHART-specific DMS objects within CHART. It provides an interface for traffic events to provide input as to what each traffic event desires to be on the sign via the ArbitrationQueue interface. Through the HARMessageNotifier interface, a HAR can use a DMS to notify travelers to tune in to a radio station to hear a traffic message. CHART business rules include concepts such as shared resources, arbitration queues, and linking device usage to traffic events. These concepts go beyond industry-standard DMS control. This includes an ability to enable and disable CHART traveler information messages, which were added in R3B3.

5.18.1.4.6 Chart2DMSConfiguration (Class)

The Chart2DMSConfiguration class is an abstract class which extends the DMSConfiguration class to provide configuration information specific to Chart II processing. Such information includes how to contact the sign under Chart II software control, the default SHAZAM message for using the sign as a HAR Notifier, and the owning organization. Such data extends beyond what would be industry-standard configuration information for a DMS. Parameters to support TCP/IP communications, notifications and more alerts, and traveler information messages were added for R3B3.

5.18.1.4.7 Chart2DMSFactory (Class)

The Chart2DMSFactory interface extends the DMSFactory interface to provide additional Chart II specific capability. This factory creates Chart2DMS objects (extensions of DMS objects). It implements the SharedResourceManager capability to control DMS objects as shared resources.

5.18.1.4.8 Chart2DMSStatus (Class)

The Chart2DMSStatus class is an abstract class which extends the DMSStatus class to provide status information specific to CHART processing, such as information on the controlling operations center for the sign. This data extends beyond what would be industry-standard status information for a DMS. Status information for traveler information messages was added in R3B3.

5.18.1.4.9 CommEnabled (Class)

The CommEnabled interface is implemented by objects that can be taken offline, put online, or put in maintenance mode through a standard interface. These states typically apply only to field devices. When a device is taken offline, it is no longer available for use through the system and automated polling (if any) is halted. When put online, a device is again available for use by TrafficEvents within the system and automated polling is enabled (if applicable). When put in maintenance mode a device is offline (i.e., cannot be used by TrafficEvents), and maintenance commands appropriate for the particular type of device are allowed to help in troubleshooting.

5.18.1.4.10 CommunicationMode (Class)

The CommunicationMode class enumerates the modes of operation for a device: ONLINE, OFFLINE, and MAINT_MODE. ONLINE is used to indicate the device is available to the operational system. OFFLINE is used to indicate the device is not available to the online system and communications to the device have been disabled. MAINT_MODE is used to indicate that the device is available only for maintenance / repair activities and testing.

5.18.1.4.11 DMS (Class)

The DMS class defines an interface to be used in manipulating Dynamic Message Sign (DMS) objects within Chart II. It specifies methods for setting messages and clearing messages from a sign (in maintenance mode), polling a sign, changing the configuration of a sign, and resetting a sign. (Setting messages on a sign in online mode are not accomplished by manipulating a DMS directly; that is accomplished by manipulating traffic events, which use an ArbitrationQueue interface or by manipulating HARs, which use a HARMessageNotifier interface. This activity involves the DMS extension, Chart2DMS, which defines interactions with signs under Chart II business rules.)

5.18.1.4.12 DMSArbQueueEntry (Class)

The DMSArbQueueEntry class provides an implementation of ArbQueueEntry that is used

for most standard entries placed on the arbitration queue. When its setActive, setInactive, and setFailed methods are called, it adds a log entry to its traffic event and calls the appropriate method on its response plan item (setActive, setInactive, or update).

5.18.1.4.13DMSConfiguration (Class)

The DMSConfiguration class is an abstract valuetype class which describes the configuration of a DMS device. This configuration information is normally fairly static: things like the size of the sign in characters and pixels, its name and location, and how to contact the sign (as opposed to dynamic information like the current message on the sign, which is defined in an analogous Status object). The font number and line spacing were added for R3B3, and the location was changed to a ObjectLocation, which contains more detailed locations fields.

5.18.1.4.14DMSConfigurationEventInfo (Class)

The DMSConfigurationEventInfo class is the type of DMSEvent used for DMSEventType DMSConfigChanged. It contains a DMSConfiguration object which details the new configuration for a Chart II DMS object.

5.18.1.4.15DMSEvent (Class)

The DMSEvent class is a union which can be any one of four events relating to DMS operations which can be pushed on an Event Channel to update event consumers on DMS-related activities. The four types of events, defined by the enumeration DMSEventType, are: DMSAdded, DMSDeleted, CurrentDMSStatus, and DMSConfigChanged.

5.18.1.4.16DMSEventType (Class)

The DMSEventType is an enumeration which defines the five types of events relating to DMS operations which can be pushed on an Event Channel to update event consumers on DMS-related activities. The five types of events are: DMSAdded, DMSDeleted, CurrentDMSStatus, DMSConfigChanged, and DMSTravInfoMsgCfgChanged.

5.18.1.4.17DMSFactory (Class)

The DMSFactory class specifies the interface to be used to create DMS objects within the Chart II system. It also provides a method to get a list of DMS devices currently in the system.

5.18.1.4.18DMSInfo (Class)

This is a structure which contains all information about a DMS: its ID, its configuration and status, the DMS type (internal or external), and a CORBA reference to the DMS.

5.18.1.4.19DMSList (Class)

The DMSList class is simply a list of DMS devices which can be used by the DMS Factory and other classes for maintaining the list or other lists of DMS objects.

5.18.1.4.20DMSMessage (Class)

The DMSMessage class is an abstract class which describes a message for a DMS. It consists of two elements: a MULTI-formatted message and beacon state information (whether the message requires that the beacons be on). The DMSMessage is contained within a DMSStatus object, used to communicate the current message on a sign, and is stored within a DMSRPIData object, used to specify the message which should be on a sign when the response plan item is executed.

5.18.1.4.21DMSModelID (Class)

The DMSModelID class enumerates the models of DMSs that are in the system.

5.18.1.4.22DMSPlanItemData (Class)

The DMSPlanItemData class is a valuetype that contains data stored in a plan item for a DMS. It is derived from PlanItemData.

5.18.1.4.23DMSRPIData (Class)

The DMSRPIData class is an abstract class which describes a response plan item for a DMS. It contains the unique identifier of the DMS to contain the DMSMessage, and the DMSMessage itself.

5.18.1.4.24DMSStatus (Class)

The DMSStatus class is an abstract value-type class which provides status information for a DMS. This status information is relatively dynamic: things like the current message on the sign, its beacon state, its current operational mode (online, offline, maintenance mode), and current operational status (OK, COMM_FAILURE, or HARDWARE_FAILURE). (More static information about the sign, such as its size and location, is defined in an analogous Configuration object.)

5.18.1.4.25DMSStatusEventInfo (Class)

The DMSStatusEventInfo class is the type of DMSEvent used for DMSEventType CurrentDMSStatus. It contains a DMSStatus object which details the new status for a Chart II DMS object.

5.18.1.4.26DMSTravInfoMsg (Class)

This class holds information necessary to put traveler information messages (containing travel times and/or toll rates) on DMSs. Each TravelerInfoMsg contains the ID for the template, and the IDs of the routes to use, as configured for its specific DMS. Each TravelerInfoMsg can be enabled or disabled. The DMSControlModule will ensure that a maximum of one TravelerInfoMsg is enabled at a time.

5.18.1.4.27DMSTravInfoMsgConfig (Class)

This class is a part of Chart2DMSConfiguration. This class holds information necessary to put traveler information messages (containing travel times and/or toll rates) on DMSs. Each DMSTravelerInfoMsgConfig contains travelTimeQueueLevel, tollRateQueueLevel, array of relatedRoutes, array of TravInfoMsg, overrideDefaultSchedule, enabledSpecificTime, and array of customSchedule.

5.18.1.4.28DMSTravInfoMsgConfigEventInfo (Class)

The DMSTravInfoMsgConfigEventInfo class is the type of DMSEvent used for DMSEventType DMSTravInfoMsgCfgChange. It contains a DMSTravInfoMsgConfig and Identifier of DMS object.

5.18.1.4.29DMSTravInfoMsgState (Class)

This enumeration lists possible states for traveler information messages. The first state is the normal case -- all others are reasons why a traveler information message may be not displayed (or not displayed correctly). It is possible that some of these states could occur at the same time, but since this is not expected to occur too often, only one state will be provided in the status. When one problem state is corrected, the next problem state (if any) would bubble up into the status. The problem states are listed in roughly priority order (although the implementation is not obliged to abide by this order if another ordering is determined to be better).

5.18.1.4.30DMSTravInfoMsgStatus (Class)

This structure provides a textual and encoded view into what is happening with the traveler information message for this DMS.

5.18.1.4.31DMSType (Class)

This is an enumeration which lists the possible types of DMS: CHART (internal to CHART) or External (imported).

5.18.1.4.32EntryOwner (Class)

Interface which must be implemented by any class which is responsible for putting an ArbQueueEntry on a device's arbitration queue. This validate method of this interface can be called by the device to determine continued validity of the entry (either during recovery or as a final check of the validity of an entry before putting its message on the device).

5.18.1.4.33ExternalDMS (Class)

The ExternalDMS class extends the DMS interface and defines a more detailed interface to be used in manipulating the External DMS objects within CHART.

5.18.1.4.34ExternalDMSFactory (Class)

The ExternalDMSFactory interface extends the DMSFactory interface.. This factory creates ExternalDMS objects (extensions of DMS objects).

5.18.1.4.35FontMetrics (Class)

The FontMetrics class is a non-behavioral class (structure) which contains information regarding to the font size used on a DMS. It is a part of a DMSConfiguration object.

5.18.1.4.36FP9500DMS (Class)

The FP9500DMS class extends the Chart2DMS interface and defines a more detailed interface to be used in manipulating FP9500 models of DMS signs. It is exemplary of potentially a whole suite of subclasses specific to a specific brand and model of sign for manufacturer-specific DMS control. For instance, the FP9500DMS has a performPixelTest method, which knows how to invoke and interpret a pixel test as supported by the FP9500 model DMS.

5.18.1.4.37FP9500DMSStatus (Class)

The FP9500DMSStatus class provides additional storage for status information unique to the FP9500 model of sign. It is exemplary of potentially a whole suite of Chart2DMSStatus subclasses specific to a specific brand and model of sign.

5.18.1.4.38GeoLocatable (Class)

This interface is implemented by objects that can provide location information to their users.

5.18.1.4.39HARMessageNotifier (Class)

The HARMessageNotifier class specifies an interface to be implemented by devices that can be used to notify the traveler to tune in to a radio station to hear a traffic message being broadcast by a HAR. A HARMessageNotifier is directional and allows users of the device to better determine if activation of the device is warranted for the message being broadcast by the HAR. This interface can be implemented by SHAZAM devices and by DMS devices which are allowed to provide a SHAZAM-like message.

5.18.1.4.40HARNotifierArbQueueEntry (Class)

The HarNotifierArbQueueEntry class provides an implementation of the ArbQueueEntry used for entries that are placed on the arbitration queue to put a "SHAZAM" message on a DMS. These types of messages have a low priority and are not allowed to overwrite any standard message (from a DMSArbQueueEntry) that is currently displayed on a device. These types of messages are also different in that they are not added to the queue directly by a response plan item and are instead included as a sub-task of activating a message on a HAR. The HAR uses a command status object to track the progress of the HAR notifier message.

5.18.1.4.41 HHMMRange (Class)

This structure defines a time duration.

5.18.1.4.42 IPPortLocationData (Class)

this structure defines the connection information of a tcp/ip port.

5.18.1.4.43 Message (Class)

This class represents a message that will be used while activating devices. This class provides a means to check if the message contains any banned words given a Dictionary object. Derived classes extend this class to provide device specific message data.

5.18.1.4.44 MessageQueue (Class)

This class represents a message queue object. It will provide the ability to add, remove, and reprioritize traffic event entries in a prioritized list.

5.18.1.4.45 MULTIParseFailure (Class)

The MULTIParseFailure class is an exception to be thrown when a MULTI-formatted DMS message cannot be correctly parsed.

5.18.1.4.46 MULTIStrIng (Class)

The MULTIStrIng class is a MULTI-formatted DMS message. The DMSMessage class contains a MULTIStrIng value to specify the content of the sign, in addition to the beacon state value.

5.18.1.4.47 NetworkConnectionSite (Class)

The NetworkConnectionSite class contains a string that is used to specify where a service is running. This field is useful for administrators in debugging problems should an object become "software comm failed".. It is included in the Chart2DMSSStatus.

5.18.1.4.48 OperationalStatus (Class)

The OperationalStatus class enumerates the types of operational status a device can have: OK (normal mode), COMM_FAILURE (no communications to the device), or HARDWARE_FAILURE (device is reachable but is reporting a hardware failure).

5.18.1.4.49 PlanItemData (Class)

This class is a valuetype that is the base class for data stored in a plan item. Derived classes contain specific data that map a device to an operation and the data needed for the operation. For example a derived class provides a mapping between a specific DMS and a DMSMessage.

5.18.1.4.50ResponsePlanItemData (Class)

This class is a delegate used to perform the execute and remove tasks for the response plan item. Derived classes of this base class have specific implementations for the type of device the response plan item is used to control.

5.18.1.4.51ResponsePlanItemTarget (Class)

This interface represents an object that can be a target of a response plan item.

5.18.1.4.52SharedResource (Class)

The SharedResource interface is implemented by any object that may have an operations center responsible for the disposition of the resource while the resource is in use.

5.18.1.4.53SharedResourceManager (Class)

The SharedResourceManager interface is implemented by classes that manage shared resources. Implementing classes must be able to provide a list of all shared resources under their management. Implementing classes must also be able to tell others if there are any resources under its management that are controlled by a given operations center. The shared resource manager is also responsible for periodically monitoring its shared resources to detect if the operations center controlling a resource doesn't have at least one user logged into the system. When this condition is detected, the shared resource manager must push an event on the ResourceManagement event channel to notify others of this condition.

5.18.1.4.54ShortErrorStatus (Class)

The ShortErrorStatus class identifies an error condition for a DMS. It is a bit field defined by the NTCIP center to field standard for DMS that specifies error conditions that may be present on the device. This class is used to encapsulate the bit mask and provide a user friendly interface to the error conditions. The DMSStatus class contains a value of this type.

5.18.1.4.55SignMetrics (Class)

The SignMetrics class is a non-behavioral class (structure) which contains information regarding to the size of a DMS, in pixels and characters. It is a part of a DMSConfiguration object.

5.18.1.4.56SignType (Class)

The SignType class defines the sign type for a DMS. Its values are defined by the SignTypeValues class. It is a part of a DMSConfiguration object.

5.18.1.4.57SignTypeValues (Class)

The SignTypeValues class enumerates the various sign types DMS devices. Examples are bos, cms, vmsChar, etc.

5.18.1.4.58TravelRouteConsumer (Class)

This interface allows other CHART objects to register as a direct consumer of travel route statistical data. It provides operations for the travel route to call when the travel time or toll rate for the route is updated. A DMS registers as a TravelRouteConsumer when a TravelerInfoMsg is enabled.

5.18.1.4.59UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.18.1.5 DeviceManagement (Class Diagram)

This class diagram shows device interfaces that are common among devices.

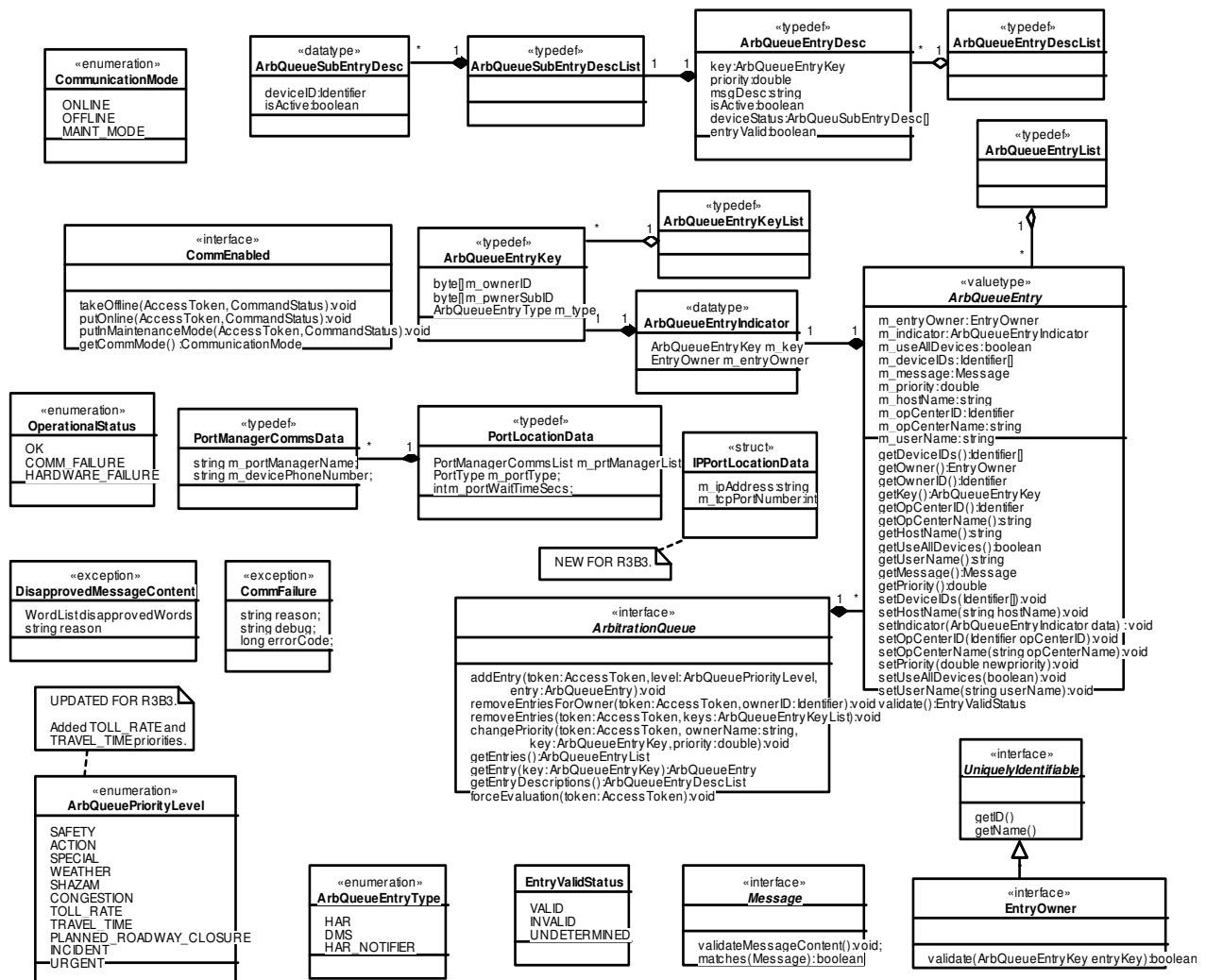


Figure 5-223. DeviceManagement (Class Diagram)

5.18.1.5.1 ArbitrationQueue (Class)

An ArbitrationQueue is a queue that arbitrates the usage of a device. The evaluation of the queue determines which message(s) should be on the device, based upon the priority of the queue entries. When entries are added to the queue, they are assigned a priority level based on the type of traffic event with which they are associated, and also upon the current contents of the queue. The priority of the queue entries can be modified after they are added to the queue. The queue is evaluated when the device is online and queue entries are added or removed, when an entry's priority is modified, or when the device is put online.

5.18.1.5.2 ArbQueueEntry (Class)

This class is used for an entry on the arbitration queue, for a single message, and for a single traffic event. (It is possible, in the case of HARNotifierArbQueueEntry objects, that certain ArbQueueEntries can be on behalf of multiple TrafficEvents. In such cases, one TrafficEvent among all those involved is picked to be the responsible TrafficEvent stored in m_indicator, the ArbQueueEntryIndicator for the entry.)

5.18.1.5.3 ArbQueueEntryDesc (Class)

This structure is used to provide a description of an entry on the arbitration queue.

5.18.1.5.4 ArbQueueEntryDescList (Class)

Collection of ArbQueueEntryDesc objects.

5.18.1.5.5 ArbQueueEntryIndicator (Class)

The ArbQueueEntryIndicator contains data necessary to specify a unique ArbQueueEntry object; in addition, it contains a reference to the TrafficEvent which is responsible for the entry.

5.18.1.5.6 ArbQueueEntryKey (Class)

This class contains the Traffic Event ID and RPI ID and is used to identify a specific ArbQueueEntry. In some cases (e.g., for HARNotifierArbQueueEntry objects), the RPI ID is the string representing a null Identifier.

5.18.1.5.7 ArbQueueEntryKeyList (Class)

A collection of ArbQueueEntryKey objects.

5.18.1.5.8 ArbQueueEntryList (Class)

5.18.1.5.9 ArbQueueEntryType (Class)

Enumeration of all possible types of entries that could be on an arbitration queue.

5.18.1.5.10ArbQueuePriorityLevel (Class)

Enumeration of all possible priority levels of the arbitration queue. All entries in the queue fit into one of these levels. The levels are named after the types of messages that are typically mapped into them. However, any message can exist in any level and new types of messages could be mapped into these levels. Thus, the levels could have been named, LOWEST through HIGHEST. The names chosen have been used to provide some indication of the likely usage of the levels. TOLL_RATE and TRAVEL_TIME levels have been added for R3B3.

5.18.1.5.11ArbQueueSubEntryDesc (Class)

This structure hold ArbQueueEntry "device-level detail for one "sub-device (such as a constituent HAR within a SyncHAR). It holds the ID of the device and an indication as to whether the entry is active for this particular subdevice. An ArbQueueEntry for a conglomerate device (such as a SyncHAR) will contain a list of these structures, one for each constituent HAR the entry is destined for.

5.18.1.5.12ArbQueueSubEntryDescList (Class)

This is an array of ArbQueueSubEntryDesc. It holds a list of "sub-devices" (such as constituent HARs of a SyncHAR) for which an arb queue entry is destined, and for which of those devices the entry is active.

5.18.1.5.13CommEnabled (Class)

The CommEnabled interface is implemented by objects that can be taken offline, put online, or put in maintenance mode through a standard interface. These states typically apply only to field devices. When a device is taken offline, it is no longer available for use through the system and automated polling (if any) is halted. When put online, a device is again available for use by TrafficEvents within the system and automated polling is enabled (if applicable). When put in maintenance mode a device is offline (i.e., cannot be used by TrafficEvents), and maintenance commands appropriate for the particular type of device are allowed to help in troubleshooting.

5.18.1.5.14CommFailure (Class)

This exception is to be thrown when an error is detected connecting to or communicating with a device.

5.18.1.5.15CommunicationMode (Class)

The CommunicationMode class enumerates the modes of operation for a device: ONLINE, OFFLINE, and MAINT_MODE. ONLINE is used to indicate the device is available to the operational system. OFFLINE is used to indicate the device is not available to the online system and communications to the device have been disabled. MAINT_MODE is used to indicate that the device is available only for maintenance / repair activities and testing.

5.18.1.5.16DisapprovedMessageContent (Class)

This exception is thrown when a text message to be put on a device contains words that are not approved. This exception is also thrown if an attempt is made to put the device in an invalid display state, such as putting the Beacons ON for a blank DMS.

5.18.1.5.17EntryOwner (Class)

Interface which must be implemented by any class which is responsible for putting an ArbQueueEntry on a device's arbitration queue. This validate method of this interface can be called by the device to determine continued validity of the entry (either during recovery or as a final check of the validity of an entry before putting its message on the device).

5.18.1.5.18EntryValidStatus (Class)

This enumeration is used to track whether an arb queue entry has been validated by its EntryOwner (interface). The possible values are VALID, INVALID, and UNDETERMINED.

5.18.1.5.19IPPortLocationData (Class)

This structure defines the connection information of a tcp/ip port.

5.18.1.5.20Message (Class)

This class represents a message that will be used while activating devices. This class provides a means to check if the message contains any banned words given a Dictionary object. Derived classes extend this class to provide device specific message data.

5.18.1.5.21OperationalStatus (Class)

The OperationalStatus class enumerates the types of operational status a device can have: OK (normal mode), COMM_FAILURE (no communications to the device), or HARDWARE_FAILURE (device is reachable but is reporting a hardware failure).

5.18.1.5.22PortLocationData (Class)

This class contains configuration data that specifies the communication server(s) to use to communicate with a device.

m_commsData - One or more objects identifying the communications server (PortManager) to use to communicate with the device, in order of preference.

m_portType - The type of port to use to communicate with the device (ISDN modem, POTS modem, direct, etc.)

m_portWaitTimeSecs - The maximum number of seconds to wait when attempting to acquire a port from a port manager.

5.18.1.5.23PortManagerCommsData (Class)

This class contains values that identify a port manager and the phone number to dial to access a device from the given port manager. This class exists to allow for the phone number used to access a device to differ based on the port manager to take into account the physical location of the port manager within the telephone network. For example, when dialing a device from one location the call may be long distance but when dialing from another location the call may be local.

5.18.1.5.24UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.18.1.6 ExternalDMS (Class Diagram)

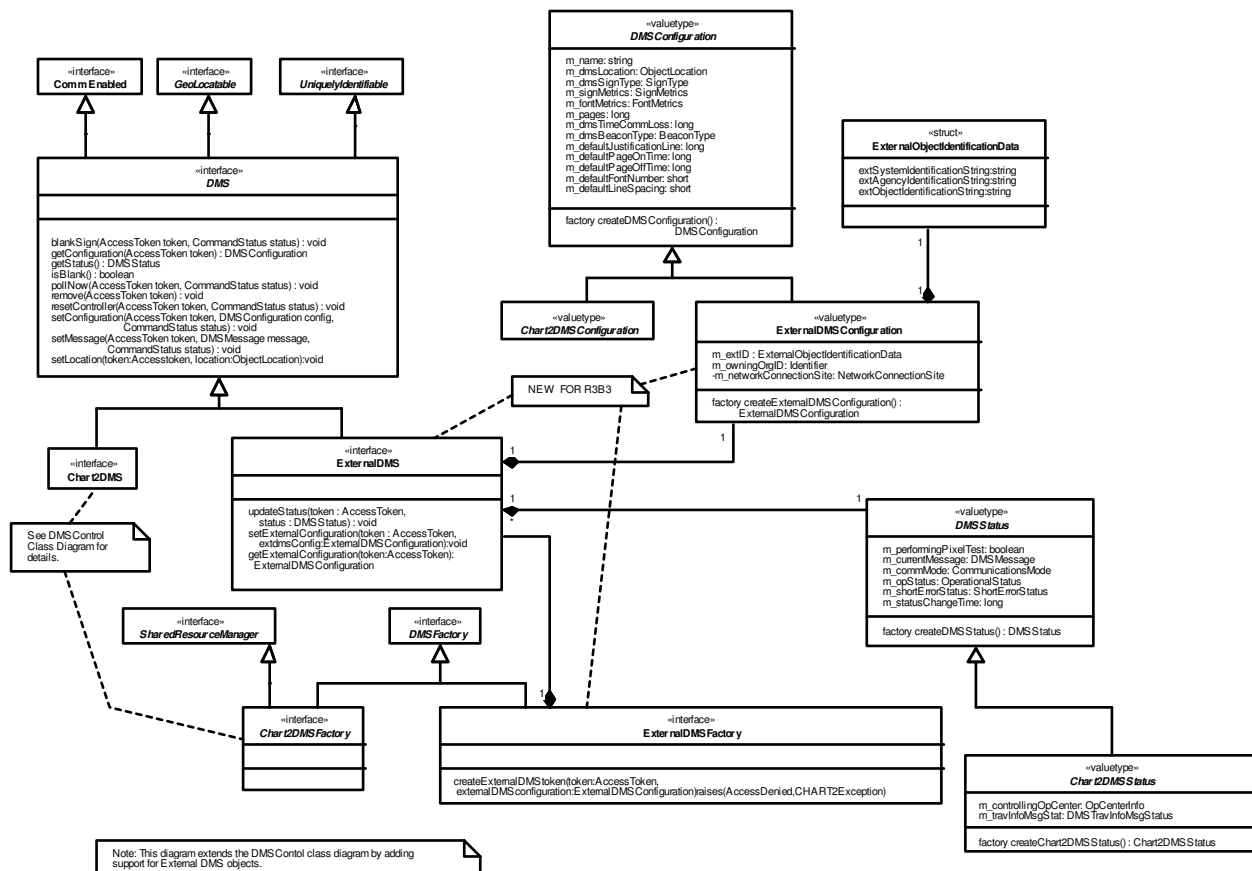


Figure 5-224. ExternalDMS (Class Diagram)

5.18.1.6.1 Chart2DMS (Class)

The Chart2DMS class extends the DMS interface and defines a more detailed interface to be used in manipulating the CHART-specific DMS objects within CHART. It provides an interface for traffic events to provide input as to what each traffic event desires to be on the sign via the ArbitrationQueue interface. Through the HARMessageNotifier interface, a HAR can use a DMS to notify travelers to tune in to a radio station to hear a traffic message. CHART business rules include concepts such as shared resources, arbitration queues, and linking device usage to traffic events. These concepts go beyond industry-standard DMS control. This includes an ability to enable and disable CHART traveler information messages, which were added in R3B3.

5.18.1.6.2 Chart2DMSConfiguration (Class)

The Chart2DMSConfiguration class is an abstract class which extends the DMSConfiguration class to provide configuration information specific to Chart II processing. Such information includes how to contact the sign under Chart II software control, the default SHAZAM message for using the sign as a HAR Notifier, and the owning organization. Such data extends beyond what would be industry-standard configuration information for a DMS. Parameters to support TCP/IP communications, notifications and more alerts, and traveler information messages were added for R3B3.

5.18.1.6.3 Chart2DMSFactory (Class)

The Chart2DMSFactory interface extends the DMSFactory interface to provide additional Chart II specific capability. This factory creates Chart2DMS objects (extensions of DMS objects). It implements the SharedResourceManager capability to control DMS objects as shared resources.

5.18.1.6.4 Chart2DMSStatus (Class)

The Chart2DMSStatus class is an abstract class which extends the DMSStatus class to provide status information specific to CHART processing, such as information on the controlling operations center for the sign. This data extends beyond what would be industry-standard status information for a DMS. Status information for traveler information messages was added in R3B3.

5.18.1.6.5 CommEnabled (Class)

The CommEnabled interface is implemented by objects that can be taken offline, put online, or put in maintenance mode through a standard interface. These states typically apply only to field devices. When a device is taken offline, it is no longer available for use through the system and automated polling (if any) is halted. When put online, a device is again available for use by TrafficEvents within the system and automated polling is enabled (if applicable). When put in maintenance mode a device is offline (i.e., cannot be used by TrafficEvents), and maintenance commands appropriate for the particular type of device are allowed to help in troubleshooting.

5.18.1.6.6 DMS (Class)

The DMS class defines an interface to be used in manipulating Dynamic Message Sign (DMS) objects within Chart II. It specifies methods for setting messages and clearing messages from a sign (in maintenance mode), polling a sign, changing the configuration of a sign, and resetting a sign. (Setting messages on a sign in online mode are not accomplished by manipulating a DMS directly; that is accomplished by manipulating traffic events, which use an ArbitrationQueue interface or by manipulating HARs, which use a HARMessageNotifier interface. This activity involves the DMS extension, Chart2DMS, which defines interactions with signs under Chart II business rules.)

5.18.1.6.7 DMSConfiguration (Class)

The DMSConfiguration class is an abstract valuetype class which describes the configuration of a DMS device. This configuration information is normally fairly static: things like the size of the sign in characters and pixels, its name and location, and how to contact the sign (as opposed to dynamic information like the current message on the sign, which is defined in an analogous Status object). The font number and line spacing were added for R3B3, and the location was changed to a ObjectLocation, which contains more detailed locations fields.

5.18.1.6.8 DMSFactory (Class)

The DMSFactory class specifies the interface to be used to create DMS objects within the Chart II system. It also provides a method to get a list of DMS devices currently in the system.

5.18.1.6.9 DMSStatus (Class)

The DMSStatus class is an abstract value-type class which provides status information for a DMS. This status information is relatively dynamic: things like the current message on the sign, its beacon state, its current operational mode (online, offline, maintenance mode), and current operational status (OK, COMM_FAILURE, or HARDWARE_FAILURE). (More static information about the sign, such as its size and location, is defined in an analogous Configuration object.)

5.18.1.6.10 ExternalDMS (Class)

The ExternalDMS class extends the DMS interface and defines a more detailed interface to be used in manipulating the External DMS objects within CHART.

5.18.1.6.11 ExternalDMSConfiguration (Class)

The ExternalDMSConfiguration class is an abstract class which extends the DMSConfiguration class to provide configuration information specific to External DMS objects.

5.18.1.6.12ExternalDMSFactory (Class)

The ExternalDMSFactory interface extends the DMSFactory interface.. This factory creates ExternalDMS objects (extensions of DMS objects).

5.18.1.6.13ExternalObjectIdentificationData (Class)

This structure is used to hold data which identifies the external source of an external object which has been imported into CHART.

5.18.1.6.14GeoLocatable (Class)

This interface is implemented by objects that can provide location information to their users.

5.18.1.6.15SharedResourceManager (Class)

The SharedResourceManager interface is implemented by classes that manage shared resources. Implementing classes must be able to provide a list of all shared resources under their management. Implementing classes must also be able to tell others if there are any resources under its management that are controlled by a given operations center. The shared resource manager is also responsible for periodically monitoring its shared resources to detect if the operations center controlling a resource doesn't have at least one user logged into the system. When this condition is detected, the shared resource manager must push an event on the ResourceManagement event channel to notify others of this condition.

5.18.1.6.16UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.18.1.7 ExternalSystem (Class Diagram)

This class diagram shows the interfaces involved in importing and exporting data from external systems.

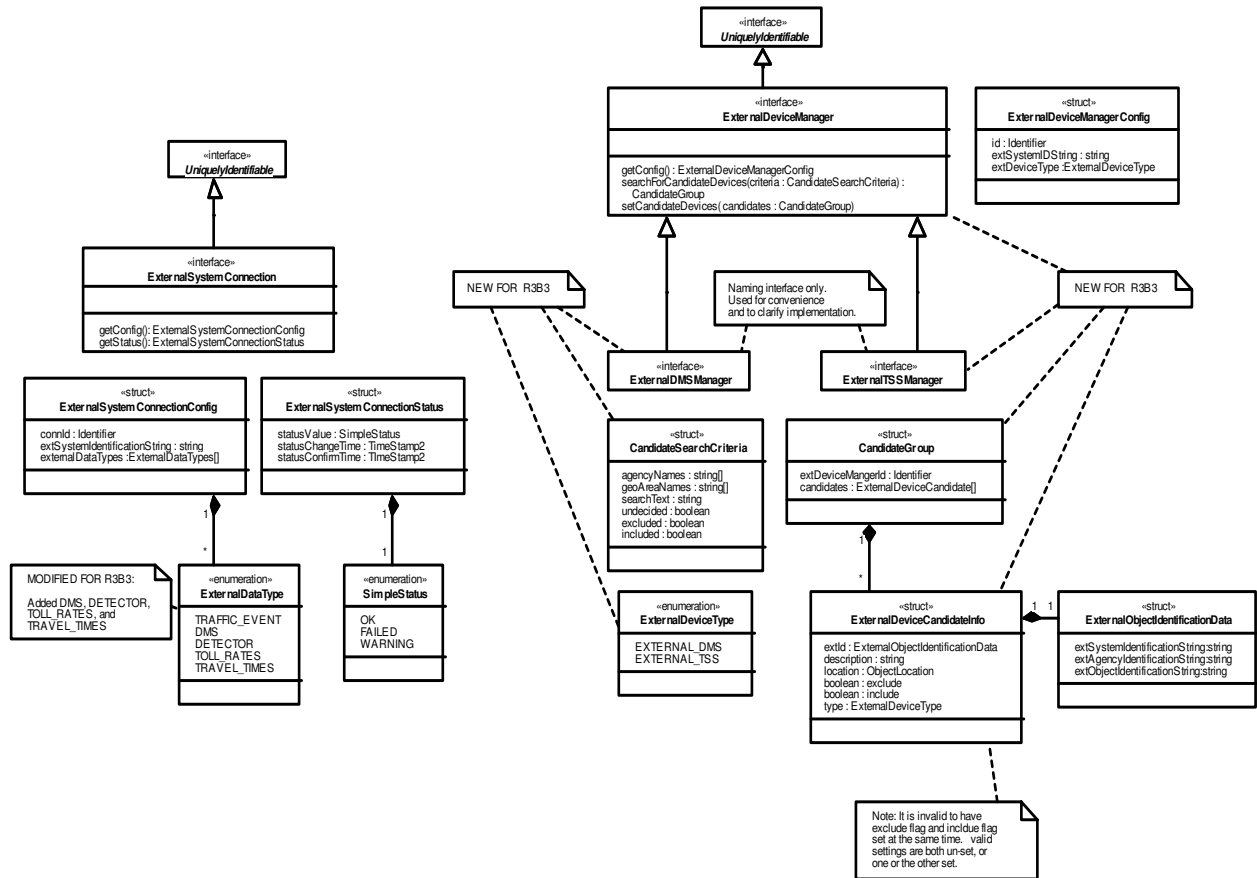


Figure 5-225. ExternalSystem (Class Diagram)

5.18.1.7.1 CandidateGroup (Class)

This struct groups **ExternalDeviceCandidates** with the **ExternalDeviceManager** that the candidates are associated with.

5.18.1.7.2 CandidateSearchCriteria (Class)

This struct is used to define search criteria when searching for candidate external devices.

5.18.1.7.3 ExternalDataType (Class)

This enumeration defines external data types handled by an **ExternalSystemConnection**.

5.18.1.7.4 ExternalDeviceCandidateInfo (Class)

This struct represents a candidate external device (DMS, TSS, ...) .

When returned from an ExternalDeviceManager.searchForCandidateDevices() query the exclude flag indicates it was previously excluded from import. The include flag indicates that it was previously included for import.

If used with a ExternalDeviceManager.setCandidates() call, it indicates whether the candidate should be included, excluded, or no action taken.

5.18.1.7.5 ExternalDeviceManager (Class)

This interface defines operations used for configuring External Devices in CHART. Each ExternalDeviceManager manages external devices of a specific type (DMS, TSS, ...) for a specific External System (i.e. data source).

5.18.1.7.6 ExternalDeviceManagerConfig (Class)

This struct defines the configuration for an ExternalDeviceManager in CHART.

5.18.1.7.7 ExternalDeviceType (Class)

This enumeration defines external device types.

5.18.1.7.8 ExternalDMSManager (Class)

This is a naming interface (empty interface) used to specify a external device manager specific to DMS devices. Its used as a convenience when querying traders and to clarify implementation.

5.18.1.7.9 ExternalObjectIdentificationData (Class)

This structure is used to hold data which identifies the external source of an external object which has been imported into CHART.

5.18.1.7.10 ExternalSystemConnection (Class)

This interface defines external connections and provides status reporting.

5.18.1.7.11 ExternalSystemConnectionConfig (Class)

This struct defines a connection to an external system.

5.18.1.7.12 ExternalSystemConnectionStatus (Class)

This struct is used to report status for an ExternalSystemConnection.

5.18.1.7.13 ExternalTSSManager (Class)

This is a naming interface (empty interface) used to specify a external device manager

specific to TSS devices. Its used as a convenience when querying traders and to clarify implementation.

5.18.1.7.14SimpleStatus (Class)

This enum defines simple status values.

5.18.1.7.15UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.18.1.8 FieldCommunications (Class Diagram)

This diagram shows system interfaces relating to field communications. These interfaces, typedefs, and enums specify the IDL for the FieldCommunications package.

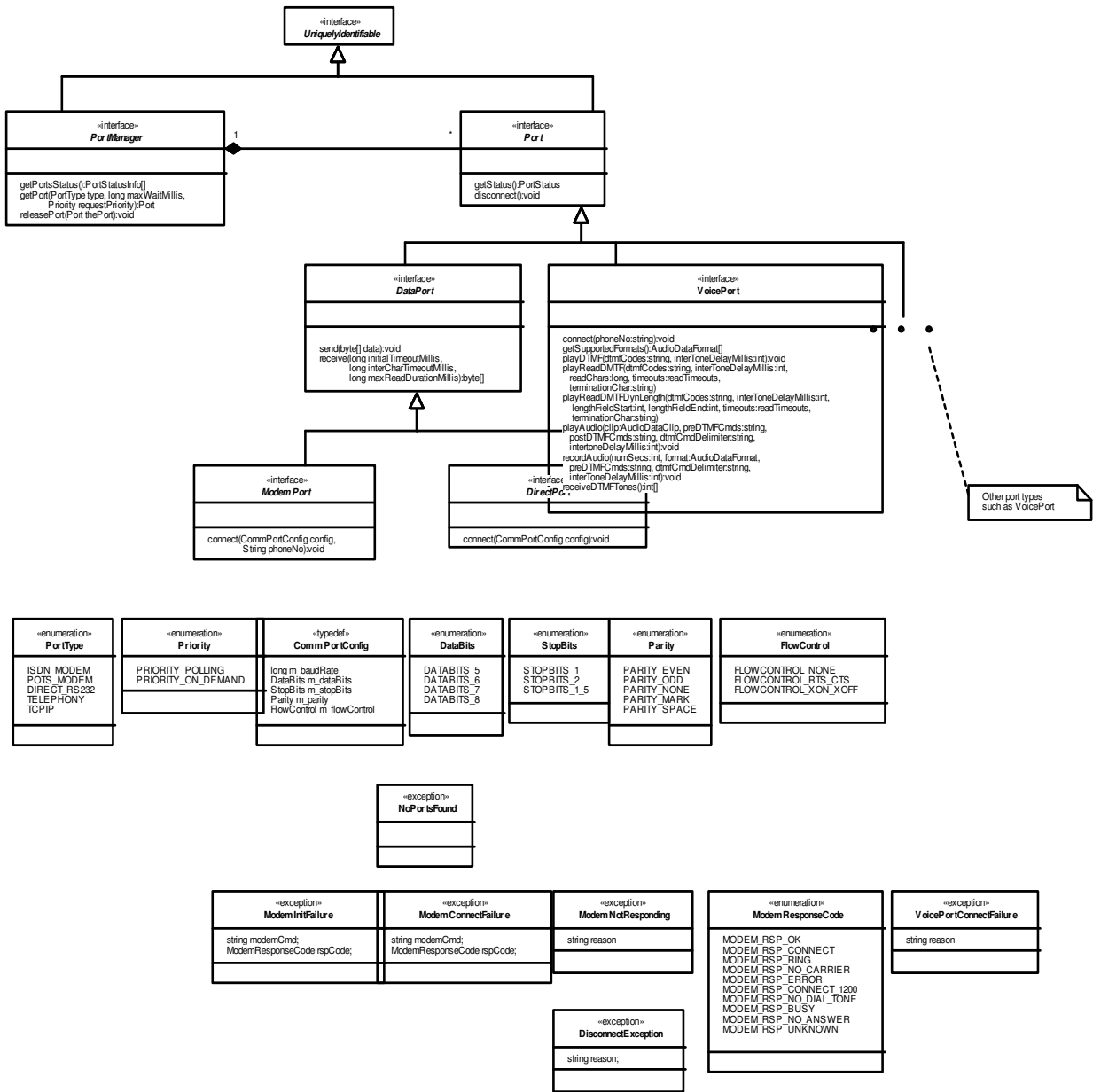


Figure 5-226. FieldCommunications (Class Diagram)

5.18.1.8.1 CommPortConfig (Class)

This structure is used to pass comm port configuration values during a connection request.

5.18.1.8.2 DataBits (Class)

This enumeration defines the valid values for data bits that may be set in a CommPortConfig structure.

5.18.1.8.3 DataPort (Class)

A DataPort is a port that allows binary data to be sent and received. Ports of this type support a receive method that allows a chunk of all available data to be received. This method prevents callers from having to issue many receive calls to parse a device response. Instead, this receive call returns all available data received within the timeout parameters. The caller can then parse the data within a local buffer. Using this mechanism, device command and response should require only one call to send and one call to receive.

5.18.1.8.4 DirectPort (Class)

A DirectPort is a Port that is directly connected to the target of communications. The connect call needs only to open the communications port.

5.18.1.8.5 DisconnectException (Class)

This exception is thrown when an error is encountered while disconnecting. There is no action that can be taken by the catch handler for this exception except to warn the user. The port will be closed and should be released as normal even if this exception is caught.

5.18.1.8.6 FlowControl (Class)

This enumeration defines the valid types of flow control that may be set in a CommPortConfig structure.

5.18.1.8.7 ModemConnectFailure (Class)

This exception is thrown when there is an error establishing a remote connection via a modem during a connection attempt on a ModemPort. This exception is generated when there is an unfavorable result to the ATDT command on the modem.

5.18.1.8.8 ModemInitFailure (Class)

This exception is thrown when there is an error initializing the modem during a connection attempt on a ModemPort.

5.18.1.8.9 ModemNotResponding (Class)

This exception is thrown when there is a failure to command a modem because the modem is not responding to commands.

5.18.1.8.10ModemPort (Class)

A ModemPort is a communications port that is capable of connecting to a remote modem. ISDN and POTS modems can be implemented under this interface.

5.18.1.8.11ModemResponseCode (Class)

This enum defines the result codes for a standard modem.

5.18.1.8.12NoPortsFound (Class)

This exception is thrown when a port is requested from a PortManager that does not have any of the requested type of port (available or in-use).

5.18.1.8.13Parity (Class)

This enumeration defines the valid values for parity that may be set in a CommPortConfig structure.

5.18.1.8.14Port (Class)

A Port is an object that models a physical communications resource. Derived interfaces specify various types of ports. All ports must be able to supply their status when requested.

5.18.1.8.15PortManager (Class)

A PortManager is an object that manages shared access to communications port resources. The getPort method is used to request the use of a port from the PortManager. Requests for ports specify the type of port needed, the priority of the request, and the maximum time the requester is willing to wait if a port is not immediately available. When the port manager returns a port, the requester has exclusive use of the port until the requester releases the port back to the PortManager or the PortManager reclaims the port due to inactivity.

5.18.1.8.16PortType (Class)

This enumeration defines the types of ports that may be requested from a PortManager.

5.18.1.8.17Priority (Class)

This enumeration specifies the priority levels used when requesting a port from a PortManager. ON_DEMAND is given higher priority than POLLING.

5.18.1.8.18StopBits (Class)

This enumeration defines the valid values for stop bits that may be set in a CommPortConfig structure.

5.18.1.8.19UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the

system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.18.1.8.20VoicePort (Class)

A voice port provides access to a port on a telephony board. It provides methods to connect it to a destination phone number and perform send and receive operations while connected that result in DTMF or voice being sent across the telephone connection to or from the device.

5.18.1.8.21VoicePortConnectFailure (Class)

This exception is thrown when the voice port fails to connect because of one of the following reasons: no dial tone, line busy or no answer.

5.18.1.9 GeoAreaManagement (Class Diagram)

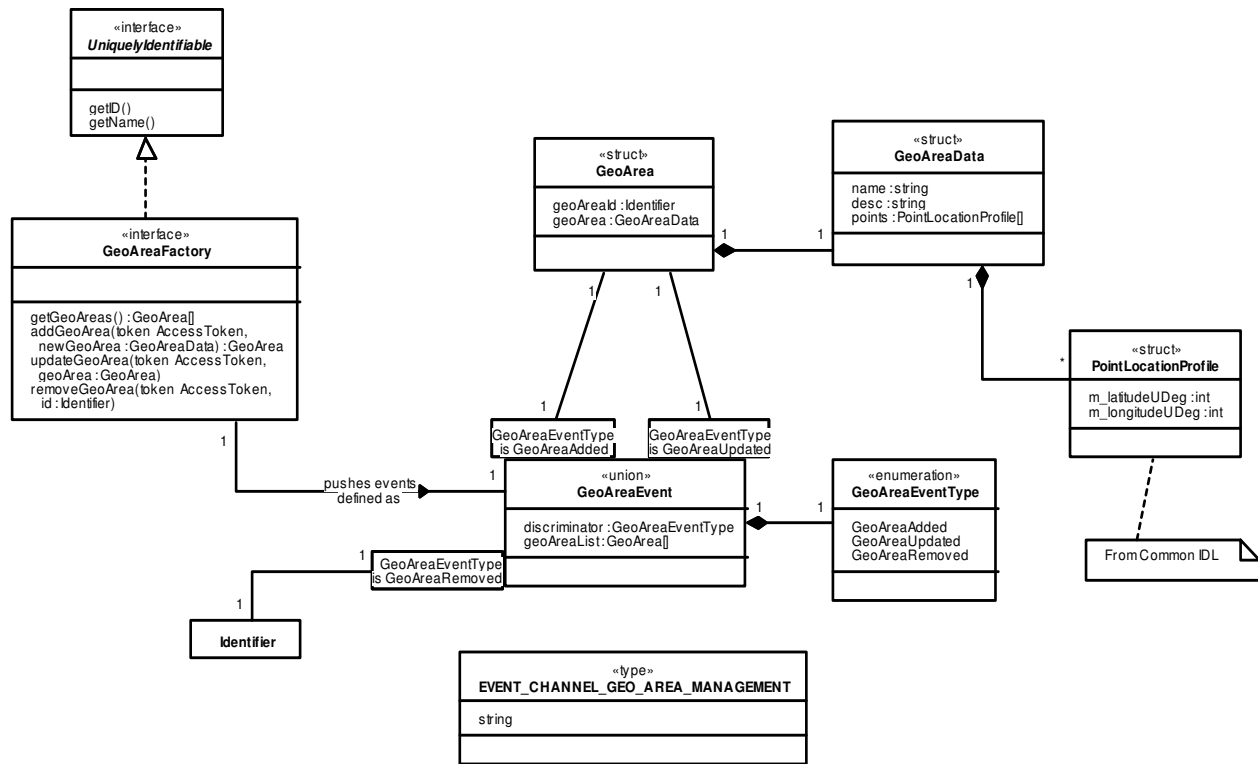


Figure 5-227. GeoAreaManagement (Class Diagram)

5.18.1.9.1 EVENT_CHANNEL_GEO_AREA_MANAGEMENT (Class)

Event Channel defined for pushing events related to Geo Area Management.

5.18.1.9.2 GeoArea (Class)

The GeoArea struct defines a unique GeoArea within the CHART system. It has a unique id and a GeoAreaData struct.

5.18.1.9.3 GeoAreaData (Class)

The GeoArea struct is a simple representation of a polygon (ordered list of points) defining a Geographical Area within the CHART system.

5.18.1.9.4 GeoAreaEvent (Class)

This class is a CORBA union that contains varying data depending on the current value of the discriminator. This union provides updates about GeoAreas.

If the discriminator is GeoAreaAdded this union contains a GeoArea object.

If the discriminator is GeoAreaUpdated this union contains a GeoArea object.

If the discriminator is GeoAreaRemoved this union contains an Common:Identifier (byte[]) object.

5.18.1.9.5 GeoAreaEventType (Class)

This enumeration defines the types of events that may be pushed on an event channel by a GeoAreaFactory object. The values in this enumeration are used as the discriminator in the GeoAreaEvent union.

5.18.1.9.6 GeoAreaFactory (Class)

This interface defines a factory responsible for managing GeoAreas with the CHART system.

5.18.1.9.7 Identifier (Class)

Wrapper class for a CHART2 identifier byte sequence. This class will be used to add identifiable objects to hash tables and perform subsequent lookup operations.

5.18.1.9.8 PointLocationProfile (Class)

This struct represents a geographical point defined as a latitude / longitude pair. The latitude and longitude are defined in microdegrees.

5.18.1.9.9 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.18.1.10 HARControl (Class Diagram)

This class diagram contains the interfaces used relating to the control of Highway Advisory Radio (HAR).

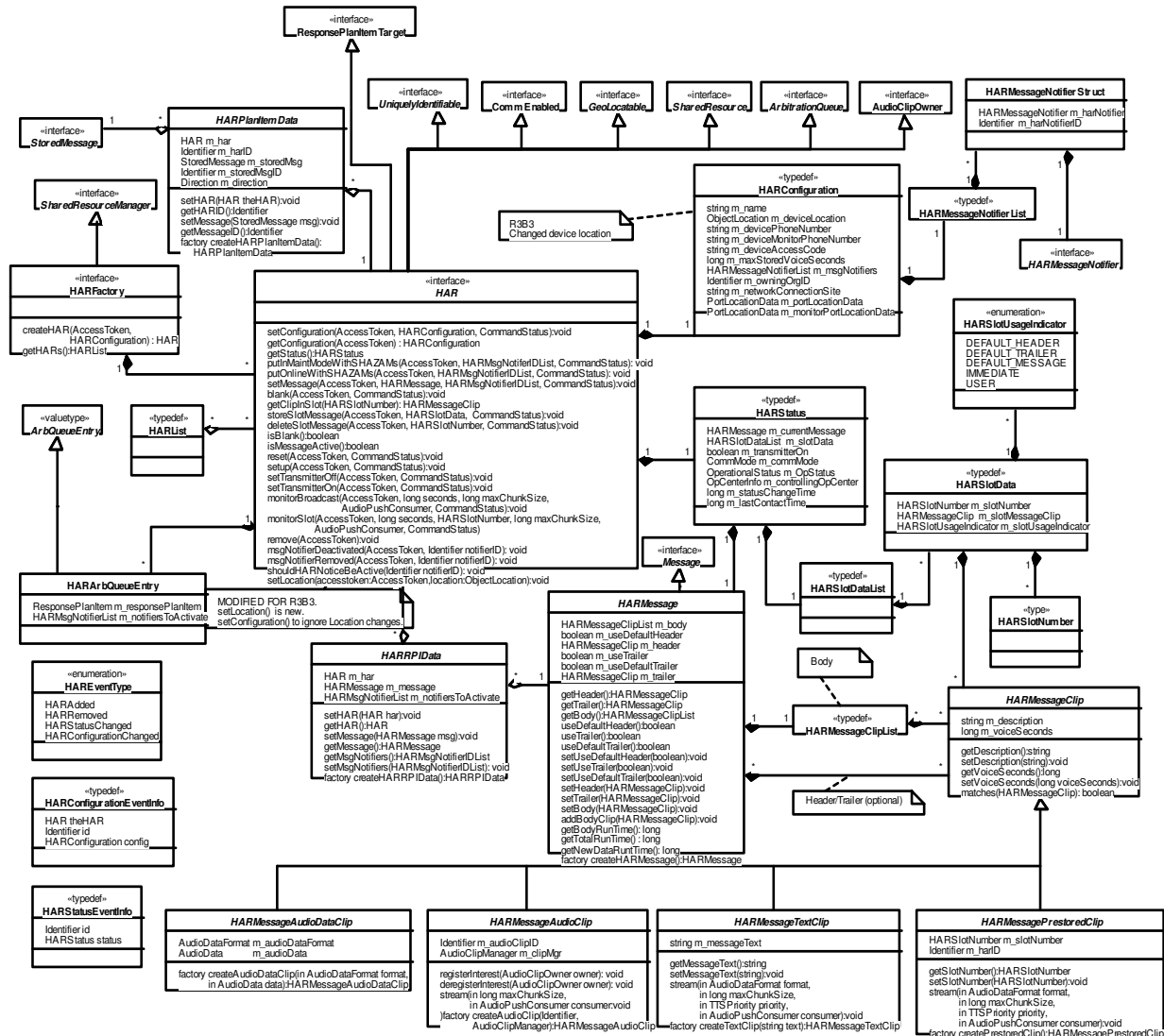


Figure 5-228. HARControl (Class Diagram)

5.18.1.10.1 ArbitrationQueue (Class)

An ArbitrationQueue is a queue that arbitrates the usage of a device. The evaluation of the queue determines which message(s) should be on the device, based upon the priority of the queue entries. When entries are added to the queue, they are assigned a priority level based on the type of traffic event with which they are associated, and also upon the current contents of the queue. The priority of the queue entries can be modified after they are

added to the queue. The queue is evaluated when the device is online and queue entries are added or removed, when an entry's priority is modified, or when the device is put online.

5.18.1.10.2ArbQueueEntry (Class)

This class is used for an entry on the arbitration queue, for a single message, and for a single traffic event. (It is possible, in the case of HARNotifierArbQueueEntry objects, that certain ArbQueueEntries can be on behalf of multiple TrafficEvents. In such cases, one TrafficEvent among all those involved is picked to be the responsible TrafficEvent stored in m_indicator, the ArbQueueEntryIndicator for the entry.)

5.18.1.10.3AudioClipOwner (Class)

This interface allows the AudioClipManager to check whether there are any parties interested in an audio clip. If no AudioClipOwners claim interest in a clip, the clip can be deleted.

5.18.1.10.4CommEnabled (Class)

The CommEnabled interface is implemented by objects that can be taken offline, put online, or put in maintenance mode through a standard interface. These states typically apply only to field devices. When a device is taken offline, it is no longer available for use through the system and automated polling (if any) is halted. When put online, a device is again available for use by TrafficEvents within the system and automated polling is enabled (if applicable). When put in maintenance mode a device is offline (i.e., cannot be used by TrafficEvents), and maintenance commands appropriate for the particular type of device are allowed to help in troubleshooting.

5.18.1.10.5GeoLocatable (Class)

This interface is implemented by objects that can provide location information to their users.

5.18.1.10.6HAR (Class)

This class is used to represent a Highway Advisory Radio (HAR) device. A HAR is used to broadcast traffic related information over a localized radio transmitter, making the information available to the traveler. This interface contains methods for getting and setting configuration, getting status, changing communications modes of a HAR, and manipulating and monitoring the HAR in maintenance and online modes.

5.18.1.10.7HARArbQueueEntry (Class)

This class is an arbitration queue entry used to set the message on a HAR on behalf of a traffic event. This entry also specifies the HARMessageNotifiers to be activated when the message is activated.

5.18.1.10.8 HARConfiguration (Class)

This class (struct) contains configuration data for a HAR device. It is used to transmit current configuration data from the HAR to the client, and to transmit proposed new configuration data from the client to the HAR. It is also used internally by the HARService to maintain its configuration in memory, and is used to transmit configuration data to/from the HAR to the HARControlDB database interface class. Device Location member has been modified for R3B3. Now it contains a detailed location information.

5.18.1.10.9 HARConfigurationEventInfo (Class)

This class defines data (HARConfiguration, and HAR ID and reference) pushed with a HARConfigurationChanged and HARAdded CORBA event.

5.18.1.10.10 HAREventType (Class)

This enumeration defines the types of CORBA events that are pushed on a HARControl event channel.

5.18.1.10.11 HARFactory (Class)

This CORBA interface allows new HAR objects to be added to the system. It also allows a requester to acquire a list of HAR objects under the domain of the specific HARFactory object.

5.18.1.10.12 HARList (Class)

The HARList class is a collection of HAR objects.

5.18.1.10.13 HARMessage (Class)

This utility class represents a message which is capable of being stored on a HAR. It stores the HAR message as a HAR message header, body and footer. The HARMessage can be configured to use the default header or can provide a custom header clip. The trailer can be specified to use the default trailer, or no trailer, or a custom trailer clip can be provided. The body can consist of one or more body clips. Users must specify one and only one body clip, but the HAR Service can combine messages for broadcast as a single combined message on a HAR, up to a maximum run length.

5.18.1.10.14 HARMessageAudioClip (Class)

This class is a thin wrapper for recorded voice that is to be played on a HAR. This class is passed around the system, wherever possible instead of passing the actual voice data contained in the initial HARMessageAudioDataClip. When the actual voice data is needed to play to the user or to program the HAR device, this object's streamer is used to stream the actual voice data back to an AudioPushConsumer specified by the requester.

5.18.1.10.15 HARMessageAudioDataClip (Class)

This class is a message clip that contains audio data and the format of the audio data. Because audio data can be very large, this type of clip is reserved for use when recorded voice is first entered into the system. Recorded voice that already exists in the system is passed throughout the system using HARMessageAudioClip to avoid sending the large audio data when possible. A HARMessageAudioClip can stream the associated data back to an audio consumer when needed, by contacting its AudioClipManager.

5.18.1.10.16 HARMessageClip (Class)

This class represents a section of a HAR message. A HARMessage typically contains one to three clips: a body plus an optional header and optional trailer. A combined HARMessage which is stored on (broadcast from) a HAR can one or more clips, an optional header, optional trailer, and one or more body clips. See HARMessage for details. A HARMessageClip can be either plain text which would need to be converted to audio prior to broadcast, or audio (WAV) format, or it can refer to a clip which is prestored in a specific target HAR already. Audio clips are normally passed around as lightweight HARMessageAudioClips, which are created from HARMessageAudioDataClips typically at the point where the HARMessageAudioClip first enters a server.

5.18.1.10.17 HARMessageClipList (Class)

The HARMessageClipList is a collection of HARMessageClip objects. It is used to specify multiple clips contained in the body of a HARMessage. While a HARMessage specified by a user can contain only one body clip, a HARMessage generated by the HAR Service can contain multiple body clips, as a result of combining more than one message into a single message for download to and broadcast by a HAR.

5.18.1.10.18 HARMessageNotifier (Class)

The HARMessageNotifier class specifies an interface to be implemented by devices that can be used to notify the traveler to tune in to a radio station to hear a traffic message being broadcast by a HAR. A HARMessageNotifier is directional and allows users of the device to better determine if activation of the device is warranted for the message being broadcast by the HAR. This interface can be implemented by SHAZAM devices and by DMS devices which are allowed to provide a SHAZAM-like message.

5.18.1.10.19 HARMessageNotifierList (Class)

This class defines a list of HARMessageNotifierStruct objects.

5.18.1.10.20 HARMessageNotifierStruct (Class)

This class (struct) defines structure used for specifying a HARMessageNotifier, containing the notifier's ID and reference.

5.18.1.10.21 HARMessagePrestoredClip (Class)

This class stores data used to identify the usage of a clip that has already been stored on a specific HAR device.

5.18.1.10.22 HARMessageTextClip (Class)

This class represents a HAR message content object which is in plain text format. This message can be checked for banned words and will be converted into a voice message using a speech engine, for downloading to a HAR device or to preview the voice audio to a user.

5.18.1.10.23 HARPlanItemData (Class)

This class is used to associate a message with a HAR for use in Plans.

5.18.1.10.24 HARRPIData (Class)

This class represents an item in a traffic event response plan that is capable of issuing a command to put a message on a HAR when executed. When the item is executed, it adds an ArbQueueEntry to the specified HAR, which stores the entry in its MessageQueue. When the item's execution is revoked, or the item is removed from the response plan (manually or implicitly through closing the traffic event) the item asks the HAR to remove the entry. The HARRPIData object also allows specification of a subset (0 to all) of the HARNotifier devices (SHAZAM or DMS devices acting as SHAZAMs) to be activated if and while the message is being broadcast on the HAR.

5.18.1.10.25 HARSlotData (Class)

This struct defines the data used to identify the contents and usage of a slot in the HAR controller.

5.18.1.10.26 HARSlotDataList (Class)

The HARSlotDataList class is simply a collection of HARSlotData objects.

5.18.1.10.27 HARSlotNumber (Class)

The HARSlotNumber is an integer used to specify slot numbers on a HAR controller.

5.18.1.10.28 HARSlotUsageIndicator (Class)

This enum defines indicators used to show the usage of a given slot in the HAR controller.

5.18.1.10.29 HARStatus (Class)

This class (struct) contains data that indicates the current status of a HAR device. The data contained in this class is that status information which can be transmitted from the HAR to the client as necessary. This struct is also used to within the HAR Service to transmit data to/from the HARControlDB database interface class. (The HAR implementation also contains other private status data elements which are not elements of this class.)

5.18.1.10.30 HARStatusEventInfo (Class)

This class contains data (HARStatus) that is pushed when the HARStatusChanged CORBA event is pushed on the HARControl event channel.

5.18.1.10.31 Message (Class)

This class represents a message that will be used while activating devices. This class provides a means to check if the message contains any banned words given a Dictionary object. Derived classes extend this class to provide device specific message data.

5.18.1.10.32 ResponsePlanItemTarget (Class)

This interface represents an object that can be a target of a response plan item.

5.18.1.10.33 SharedResource (Class)

The SharedResource interface is implemented by any object that may have an operations center responsible for the disposition of the resource while the resource is in use.

5.18.1.10.34 SharedResourceManager (Class)

The SharedResourceManager interface is implemented by classes that manage shared resources. Implementing classes must be able to provide a list of all shared resources under their management. Implementing classes must also be able to tell others if there are any resources under its management that are controlled by a given operations center. The shared resource manager is also responsible for periodically monitoring its shared resources to detect if the operations center controlling a resource doesn't have at least one user logged into the system. When this condition is detected, the shared resource manager must push an event on the ResourceManagement event channel to notify others of this condition.

5.18.1.10.35 StoredMessage (Class)

This class holds a message object that is stored in a message in a library. It contains attributes such as category and message description which are used to allow the user to organize messages.

5.18.1.10.36 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.18.1.11 HARNotification (Class Diagram)

This Class Diagram shows the classes involved in manipulating HAR message notifications. The HAR notifiers can be SHAZAMs or DMS devices that are acting as SHAZAMs. Note that R1B2 prevents a DMS SHAZAM message from overwriting another type of DMS message.

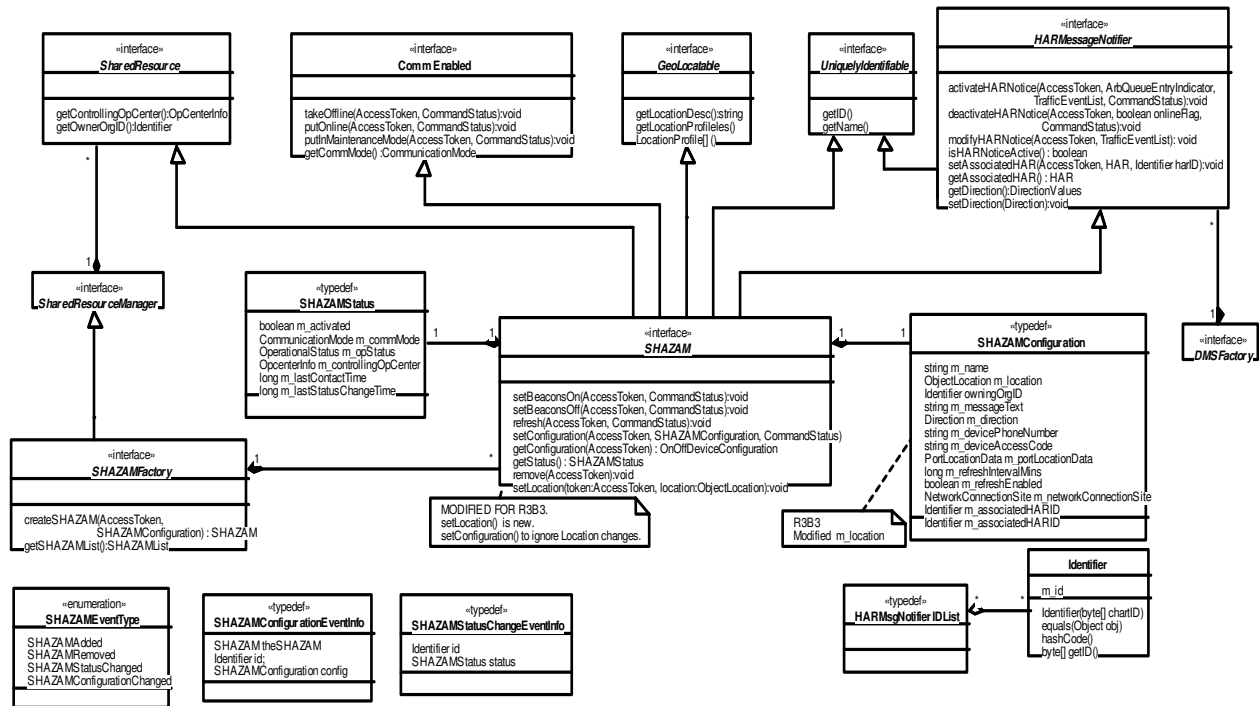


Figure 5-229. HARNotification (Class Diagram)

5.18.1.11.1 CommEnabled (Class)

The **CommEnabled** interface is implemented by objects that can be taken offline, put online, or put in maintenance mode through a standard interface. These states typically apply only to field devices. When a device is taken offline, it is no longer available for use through the system and automated polling (if any) is halted. When put online, a device is again available for use by **TrafficEvents** within the system and automated polling is enabled (if applicable). When put in maintenance mode a device is offline (i.e., cannot be used by **TrafficEvents**), and maintenance commands appropriate for the particular type of device are allowed to help in troubleshooting.

5.18.1.11.2 DMSFactory (Class)

The **DMSFactory** class specifies the interface to be used to create DMS objects within the Chart II system. It also provides a method to get a list of DMS devices currently in the system.

5.18.1.11.3GeoLocatable (Class)

This interface is implemented by objects that can provide location information to their users.

5.18.1.11.4HARMessageNotifier (Class)

The HARMessageNotifier class specifies an interface to be implemented by devices that can be used to notify the traveler to tune in to a radio station to hear a traffic message being broadcast by a HAR. A HARMessageNotifier is directional and allows users of the device to better determine if activation of the device is warranted for the message being broadcast by the HAR. This interface can be implemented by SHAZAM devices and by DMS devices which are allowed to provide a SHAZAM-like message.

5.18.1.11.5HARMsgNotifierIDList (Class)

This typedef is a sequence of HARMessageNotifier identifiers.

5.18.1.11.6Identifier (Class)

Wrapper class for a CHART2 identifier byte sequence. This class will be used to add identifiable objects to hash tables and perform subsequent lookup operations.

5.18.1.11.7SharedResource (Class)

The SharedResource interface is implemented by any object that may have an operations center responsible for the disposition of the resource while the resource is in use.

5.18.1.11.8SharedResourceManager (Class)

The SharedResourceManager interface is implemented by classes that manage shared resources. Implementing classes must be able to provide a list of all shared resources under their management. Implementing classes must also be able to tell others if there are any resources under its management that are controlled by a given operations center. The shared resource manager is also responsible for periodically monitoring its shared resources to detect if the operations center controlling a resource doesn't have at least one user logged into the system. When this condition is detected, the shared resource manager must push an event on the ResourceManagement event channel to notify others of this condition.

5.18.1.11.9SHAZAM (Class)

This interface class is used to identify the SHAZAM-specific methods which can be used to interface with a SHAZAM field device. It specifies methods for activating and deactivating the SHAZAM in maintenance mode, refreshing the SHAZAM (commanding the device to its last known status), changing the configuration of the SHAZAM, and removing the SHAZAM. This interface is implemented by a SHAZAMImpl class, which uses a helper ProtocolHdlr class to perform the model specific protocol for device command and control.

5.18.1.11.10 SHAZAMConfiguration (Class)

This class contains data that specifies the configuration of a SHAZAM device. It is used to communicate configuration information to/from the database, and to/from the GUI clients. The GUI sends a SHAZAMConfiguration when creating a SHAZAM or modifying the configuration of an existing SHAZAM. Device Location member has been modified for R3B3. Now it contains a detailed location information.

5.18.1.11.11 SHAZAMConfigurationEventInfo (Class)

This class contains data (a SHAZAMConfiguration object) that is pushed on the SHAZAMControl CORBA event channel with a SHAZAMConfigurationChanged or SHAZAMAdded event type.

5.18.1.11.12 SHAZAMEventType (Class)

This enum defines the types of CORBA events that are pushed on a SHAZAM control event channel.

5.18.1.11.13 SHAZAMFactory (Class)

The SHAZAMFactory class specifies the interface to be used to create SHAZAM objects within the Chart II system. It also provides a method to get a list of SHAZAM devices currently in the system.

5.18.1.11.14 SHAZAMStatus (Class)

This class contains the current status of a SHAZAM device. This class is used to store status within the SHAZAM object, and is also used to communicate configuration information to/from the database, and to the GUI clients (one-way).

5.18.1.11.15 SHAZAMStatusChangeEventInfo (Class)

This class contains data (a SHAZAMStatus object) that is pushed on a SHAZAMControl event channel with a SHAZAMStatusChanged event.

5.18.1.11.16 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.18.1.12 MessageTemplateManagement (Class Diagram)

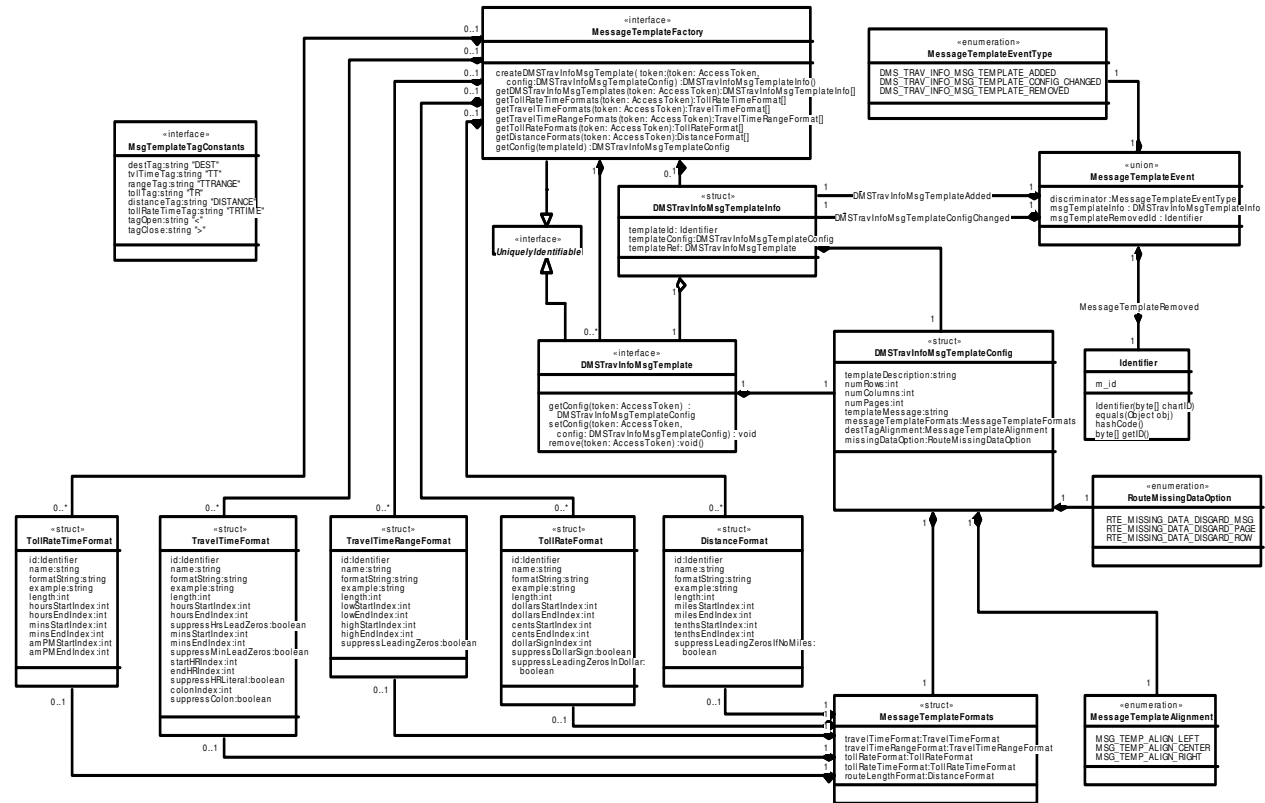


Figure 5-230. MessageTemplateManagement (Class Diagram)

5.18.1.12.1 DistanceFormat (Class)

This object contains the data for a distance format in the CHART DB.

5.18.1.12.2 DMSTravInfoMsgTemplate (Class)

The DMSTravInfoMsgTemplate interface is implemented by objects that allow execution of tasks associated with DMS travel information message templates.

5.18.1.12.3 DMSTravInfoMsgTemplateAdded (Composition)

5.18.1.12.4 DMSTravInfoMsgTemplateConfig (Class)

This object contains the configuration data for a message template that represents a DMSTravInfoMsgTemplate in the CHART DB

5.18.1.12.5DMSTravInfoMsgTemplateInfo (Class)

This struct contains a DMS travel information message configuration and a CORBA reference to the DMS travel information message template object.

5.18.1.12.6Identifier (Class)

Wrapper class for a CHART2 identifier byte sequence. This class will be used to add identifiable objects to hash tables and perform subsequent lookup operations.

5.18.1.12.7MessageTemplateAlignment (Class)

This IDL enumeration defines the message template alignment options supported in the DMSTravInfoMsgTemplateConfig. These can either be align left, align center or align right.

5.18.1.12.8MessageTemplateEvent (Class)

This union identifies the data to be passed with events that are pushed through the CORBA event service in relation to message templates.

5.18.1.12.9MessageTemplateEventType (Class)

This IDL enumeration defines the types of CORBA Events supported in the MessageTemplateModule. These can either be DMS message Created, Changed, or Removed.

5.18.1.12.10 MessageTemplateFactory (Class)

Interface whose implementation is used to create message templates, retrieve travel information message templates and retrieve travel time and toll rate formats.

5.18.1.12.11 MessageTemplateFormats (Class)

This structure contains all travel time and toll rate format that are specified within a given DMSTravInfoMsgTemplate.

5.18.1.12.12 MsgTemplateTagConstants (Class)

The MessageTemplateTagConstants interface contains constants that are used to define tags used to create DMS travel information message templates.

5.18.1.12.13 RouteMissingDataOption (Class)

This IDL enumeration defines the route missing data options supported in the DMSTravInfoMsgTemplateConfig. These can either be discard row, discard page or discard row.

5.18.1.12.14 TollRateFormat (Class)

This object contains the data for a toll rate format in the CHART DB.

5.18.1.12.15 TollRateTimeFormat (Class)

This object contains the data for a toll rate time format in the CHART DB.

5.18.1.12.16 TravelTimeFormat (Class)

This object contains the data for a travel time format in the CHART DB.

5.18.1.12.17 TravelTimeRangeFormat (Class)

This object contains the data for a travel time range format in the CHART DB.

5.18.1.12.18 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.18.1.13 ResourceManagement (Class Diagram)

This class diagram contains the interfaces pertaining to shared resources, operations centers, user login sessions, and organizations.

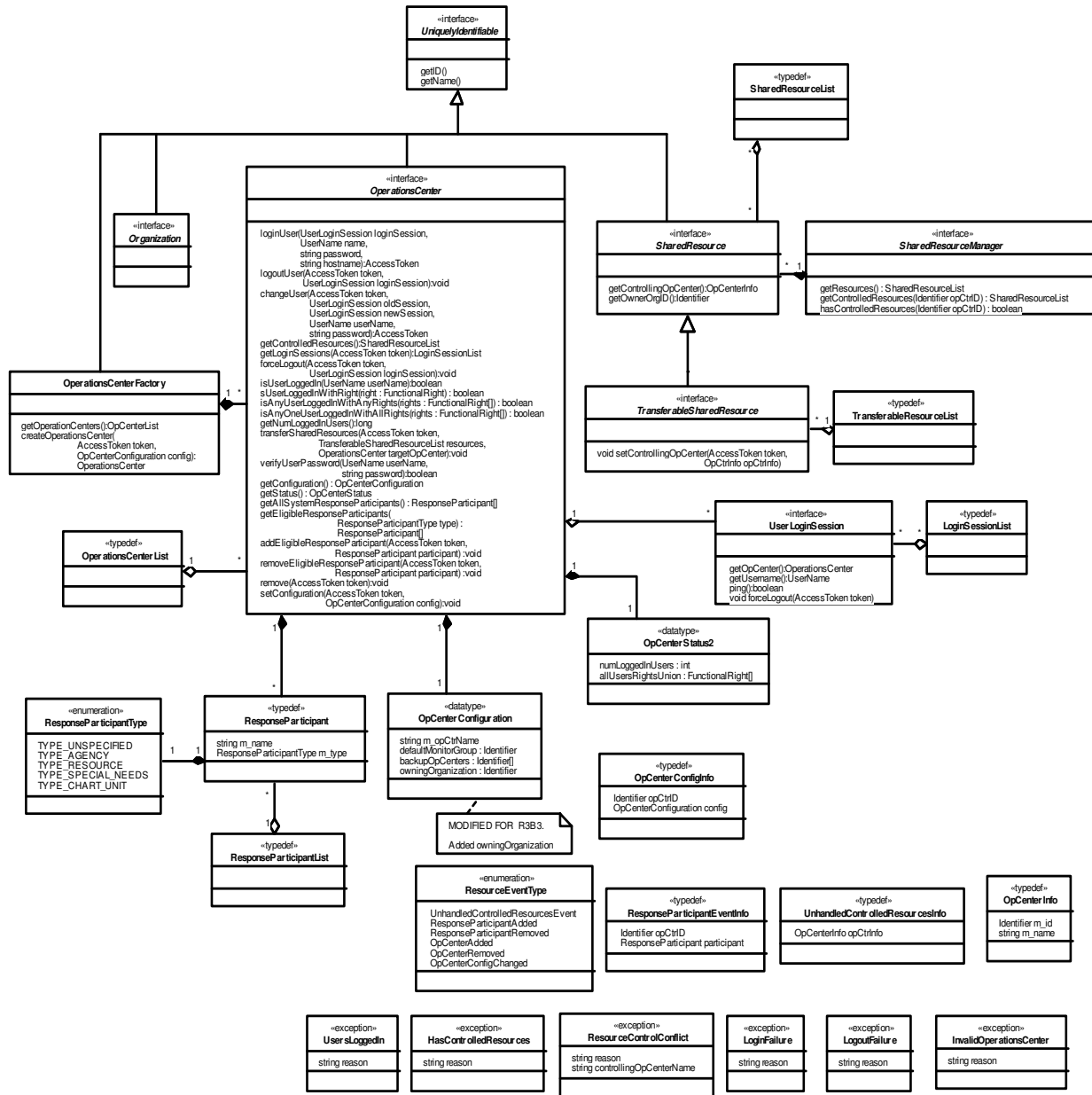


Figure 5-231. ResourceManagement (Class Diagram)

5.18.1.13.1 HasControlledResources (Class)

This class represents an exception which describes a failure caused when the user tries to do

something which requires that no resources be controlled, yet the Operations Center which the user is logged in to is still controlling one or more shared resources.

5.18.1.13.2InvalidOperationsCenter (Class)

Exception which describes a failure caused when the operations center specified is not valid for the attempted operation.

5.18.1.13.3LoginFailure (Class)

This class represents an exception which describes a login failure.

5.18.1.13.4LoginSessionList (Class)

A LoginSessionList is simply a collection of UserLoginSession objects.

5.18.1.13.5LogoutFailure (Class)

This exception is thrown when an error occurs while logging a user out of the system.

5.18.1.13.6OpCenterConfigInfo (Class)

This structure contains information pertaining to a change in the configuration of an operations center.

5.18.1.13.7OpCenterConfiguration (Class)

This structure contains the configuration data for an operations center.

5.18.1.13.8OpCenterInfo (Class)

This structure contains the information about an OperationsCenter.

5.18.1.13.9OpCenterStatus2 (Class)

The actual name of this class is OpCenterStatus. It represents the status of an operations center. This class was introduced for R3B1 to transmit status of operations centers from the Resource Manager serving it to other interested parties. The data stored in the OpCenterStatus includes the number of users currently logged into the center and the union of all functional rights held by all users currently logged into that center.

5.18.1.13.10 OperationsCenter (Class)

The OperationsCenter represents a center where one or more users are located. This class is used to log users into the system. If the username and password provided to the loginUser method are valid, the caller is given a token that contains information about the user and the functional rights of the user. This token is then used to call privileged methods within the system. Shared resources in the system are either available or under the control of an OperationsCenter. The OperationsCenter keeps track of users that are logged in so that it can ensure that the last user does not log out while there are shared resources under its

control. This list of logged in users is also available for monitoring system usage or to force users to logout for system maintenance.

5.18.1.13.11 OperationsCenterFactory (Class)

This class is used to create new operations centers and maintain them in a collection.

5.18.1.13.12 OperationsCenterList (Class)

This represents a collection of OperationsCenter objects.

5.18.1.13.13 Organization (Class)

The Organization interface extends the UniquelyIdentifiable interface and will represent an organization, that is an administrative body which can control or own resources.

5.18.1.13.14 ResourceControlConflict (Class)

This exception is thrown when attempt to gain control of a shared resource fails because the resource is under the control of a different operations center and the requesting user does not have the functional right to override the restriction.

5.18.1.13.15 ResourceEventType (Class)

The ResourceEventType enumeration defines all of the resource related event types.

5.18.1.13.16 ResponseParticipant (Class)

The ResponseParticipant class is a non-behavioral structure which specifies a participant in a response.

5.18.1.13.17 ResponseParticipantEventInfo (Class)

This structure contains information about an eligible response participant that is added to an operations center.

5.18.1.13.18 ResponseParticipantList (Class)

This represents a collection of ResponseParticipant objects.

5.18.1.13.19 ResponseParticipantType (Class)

The ResponseParticipantType enumeration defines a type of entity participating in a response to an event. This could be an external organization, a mobile unit, a mobile device or special purpose vehicle, or a special needs vehicle equipped to handle unusual or hazardous situations.

5.18.1.13.20 SharedResource (Class)

The SharedResource interface is implemented by any object that may have an operations

center responsible for the disposition of the resource while the resource is in use.

5.18.1.13.21 SharedResourceList (Class)

A SharedResourceList is simply a collection of SharedResource objects.

5.18.1.13.22 SharedResourceManager (Class)

The SharedResourceManager interface is implemented by classes that manage shared resources. Implementing classes must be able to provide a list of all shared resources under their management. Implementing classes must also be able to tell others if there are any resources under its management that are controlled by a given operations center. The shared resource manager is also responsible for periodically monitoring its shared resources to detect if the operations center controlling a resource doesn't have at least one user logged into the system. When this condition is detected, the shared resource manager must push an event on the ResourceManagement event channel to notify others of this condition.

5.18.1.13.23 TransferableResourceList (Class)

This represents a collection of transferable shared resources.

5.18.1.13.24 TransferableSharedResource (Class)

The TransferrableSharedResource interface extends the SharedResource interface, which is implemented by SharedResource objects whose control can be transferred from one operations center to another.

5.18.1.13.25 UnhandledControlledResourcesInfo (Class)

The UnhandledControlledResourcesEvent class is an event pushed when it is detected that an OperationsCenter is controlling one or more controlled resources but has no users logged in.

5.18.1.13.26 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.18.1.13.27 UserLoginSession (Class)

The UserLoginSession CORBA interface is used to store information about a user that is logged into the system. This object is served from the GUI and provides a means for the servers to call back into the GUI process.

5.18.1.13.28 UsersLoggedIn (Class)

This exception is thrown if an attempt is made to remove the operations center when users are logged in.

5.18.1.14 TSSManagement (Class Diagram)

This class diagram contains the interfaces, structs, and typedefs that are to be defined in IDL and provide the external interface to the TSSManagement package of the CHART II system.

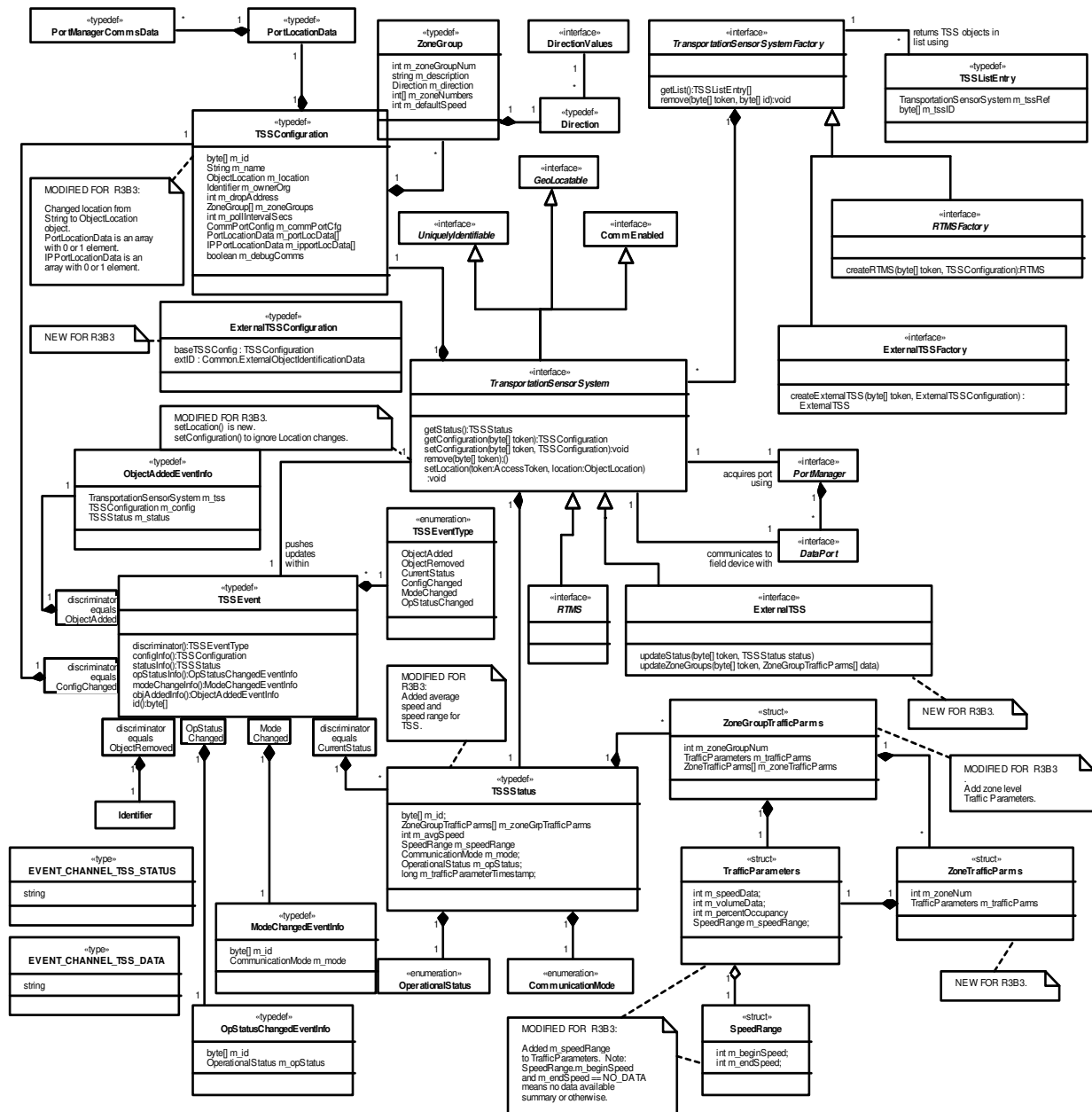


Figure 5-232. TSSManagement (Class Diagram)

5.18.1.15 CommEnabled (Class)

The CommEnabled interface is implemented by objects that can be taken offline, put online, or put in maintenance mode through a standard interface. These states typically apply only to field devices. When a device is taken offline, it is no longer available for use through the system and automated polling (if any) is halted. When put online, a device is again available for use by TrafficEvents within the system and automated polling is enabled (if applicable). When put in maintenance mode a device is offline (i.e., cannot be used by TrafficEvents), and maintenance commands appropriate for the particular type of device are allowed to help in troubleshooting.

5.18.1.16 CommunicationMode (Class)

The CommunicationMode class enumerates the modes of operation for a device: ONLINE, OFFLINE, and MAINT_MODE. ONLINE is used to indicate the device is available to the operational system. OFFLINE is used to indicate the device is not available to the online system and communications to the device have been disabled. MAINT_MODE is used to indicate that the device is available only for maintenance / repair activities and testing.

5.18.1.17 DataPort (Class)

A DataPort is a port that allows binary data to be sent and received. Ports of this type support a receive method that allows a chunk of all available data to be received. This method prevents callers from having to issue many receive calls to parse a device response. Instead, this receive call returns all available data received within the timeout parameters. The caller can then parse the data within a local buffer. Using this mechanism, device command and response should require only one call to send and one call to receive.

5.18.1.18 Direction (Class)

This type defines a short value that is used to indicate a direction of travel as defined in DirectionValues.

5.18.1.19 DirectionValues (Class)

This interface contains constants for directions as defined in the TMDD.

5.18.1.20 EVENT_CHANNEL_TSS_DATA (Class)

This is a static string that contains the name of the event channel used to push events that contain Transportation Sensor System traffic parameter data. The following TSSEventTypes are pushed on EVENT_CHANNEL_TSS_DATA channels:

CurrentStatus

5.18.1.21 EVENT_CHANNEL_TSS_STATUS (Class)

This is a static string that contains the name of the event channel used to push events relating to the change in a Transportation Sensor System status and/or configuration. The following TSSEventTypes are pushed on EVENT_CHANNEL_TSS_STATUS channels:

ObjectAdded

ObjectRemoved

ConfigChanged

ModeChanged

OpStatusChanged

5.18.1.22 ExternalTSS (Class)

This interface represents an External Systems TSS in the Chart System. I.E. a proxy for a physical TSS outside of Chart.

5.18.1.23 ExternalTSSConfiguration (Class)

This class holds configuration data for an ExternalTSS. It extends the TSSConfiguration data by including a reference to the base TSSConfig.

baseTSSConfig - Reference to the base TSSConfig.

extID - This objects holds the External System Name / Ext Agency / Ext id for this Extenral TSS. This uniquely identifies it in Chart.

5.18.1.24 ExternalTSSFactory (Class)

This interface extends the TransportationSensorSystemFactory interface to allow support of ExternalTSS objects in Chart.

5.18.1.25 GeoLocatable (Class)

This interface is implemented by objects that can provide location information to their users.

5.18.1.26 Identifier (Class)

Wrapper class for a CHART2 identifier byte sequence. This class will be used to add identifiable objects to hash tables and perform subsequent lookup operations.

5.18.1.27 ModeChangedEventInfo (Class)

This struct contains information pushed with a ModeChanged event.

m_id - The ID of the TSS whose communication mode has changed.

m_mode - The new communication mode for the TSS.

5.18.1.28 ObjectAddedEventInfo (Class)

This structure contains information passed in the ObjectAdded event pushed on a TSS status event channel. It contains the object reference that has been added along with its configuration values and current status values.

5.18.1.29 OperationalStatus (Class)

The OperationalStatus class enumerates the types of operational status a device can have: OK (normal mode), COMM_FAILURE (no communications to the device), or HARDWARE_FAILURE (device is reachable but is reporting a hardware failure).

5.18.1.30 OpStatusChangedEventInfo (Class)

This struct contains data passed with an OpStatusChanged event.

m_id - The ID of the TSS whose operational status has changed.

m_opStatus - The new operational status for the device.

5.18.1.31 PortLocationData (Class)

This class contains configuration data that specifies the communication server(s) to use to communicate with a device.

m_commsData - One or more objects identifying the communications server (PortManager) to use to communicate with the device, in order of preference.

m_portType - The type of port to use to communicate with the device (ISDN modem, POTS modem, direct, etc.)

m_portWaitTimeSecs - The maximum number of seconds to wait when attempting to acquire a port from a port manager.

5.18.1.32 PortManager (Class)

A PortManager is an object that manages shared access to communications port resources. The getPort method is used to request the use of a port from the PortManager. Requests for ports specify the type of port needed, the priority of the request, and the maximum time the requester is willing to wait if a port is not immediately available. When the port manager returns a port, the requester has exclusive use of the port until the requester releases the port

back to the PortManager or the PortManager reclaims the port due to inactivity.

5.18.1.33 PortManagerCommsData (Class)

This class contains values that identify a port manager and the phone number to dial to access a device from the given port manager. This class exists to allow for the phone number used to access a device to differ based on the port manager to take into account the physical location of the port manager within the telephone network. For example, when dialing a device from one location the call may be long distance but when dialing from another location the call may be local.

5.18.1.34 RTMS (Class)

The Remote Traffic Microwave Sensor (RTMS) is a detector manufactured by EIS, Inc. capable of providing lane level volume, speed, and occupancy data for up to 8 lanes of a roadway at a single location. This interface serves to identify TransportationSensorSystem objects as being of the type RTMS. It also provides a place holder for future operations that may not apply to TSS objects in general and are instead RTMS specific.

5.18.1.35 RTMSFactory (Class)

Objects which implement RTMSFactory are capable of adding an RTMS to the system.

5.18.1.36 SpeedRange (Class)

This struct is used to specify a speed range. The speed range is defined in MPH and has an upper and lower limit inclusive. Note: m_endSpeed of zero means range is > m_beginSpeed. MPH is implied.

5.18.1.37 TrafficParameters (Class)

This struct contains traffic parameters that are sensed and reported by a Traffic Sensor System such as the RTMS.

m_speedData - The arithmetic mean of the speeds collected over a sample period in miles per hour in tenths. (thus 550 == 55.0 MPH) Valid values are 0 to 2550. A value of 65535 is used to indicate a missing or invalid value (such as when the volume for the sample period is zero).

m_volumeData - The count of vehicles for the sample period. Valid values 0 to 65535. A value of 65535 represents a missing value.

m_percentOccupancy - The percentage of occupancy of the roadway in tenths of a percent. (thus 1000 = 100.0 percent). Valid values are 0 to 1000. A value of 65535 represents a missing or invalid value.

5.18.1.38 TransportationSensorSystem (Class)

A Transportation Sensor System (TSS) is a generic term used to describe a class of technology used for detection within the transportation industry. Examples of TSS devices range from the advanced devices, such as RTMS, to basic devices, such as single loop detectors.

This software interface is implemented by objects that provide access to the traffic parameters sensed by a Transportation Sensor System. Transportation Sensor Systems are capable of providing detection for one or more detection zones. A single loop detector would have one detection zone, while an RTMS would have 8 detection zones.

5.18.1.39 TransportationSensorSystemFactory (Class)

This interface is implemented by objects that are used to create and serve TransportationSensorSystem (TSS) Objects. All factories of TSS objects can return the list of TSS objects which they have created and serve. Derived interfaces are used to provide factories to create specific make, models, and types of TransportationSensorSystem objects.

5.18.1.40 TSSConfiguration (Class)

This class holds configuration data for a transportation sensor system (TSS) as follows:

m_id - The unique identifier for this TSS. This field is ignored when the object is passed to the TSS to change its configuration.

m_name - The name used to identify the TSS.

m_location - A descriptive location of the TSS.

m_dropAddress - The drop address for the device.

m_zoneGroups - Logical groupings of detection zones, used to provide a single set of traffic parameters for one or more detection zones.

m_pollIntervalSecs - The interval on which the TSS should be polled for its current traffic parameters (in seconds).

m_commPortCfg - Communication configuration values.

m_portLocData - Configuration information that determines which port manager(s) should be used to establish a connection with the SensorSystem.

m_debugComms - Flag used to enable/disable the logging of communications data for this TSS. When enabled, command and response packets exchanged with the device are logged to a debugging log file.

5.18.1.41 TSSEvent (Class)

This class is a CORBA union that contains varying data depending on the current value of the discriminator.

If the discriminator is ConfigChanged, this union contains a TSSConfig object.

If the discriminator is ObjectAdded, this union contains an ObjectAddedEventInfo object.

If the discriminator is ObjectRemoved, this union contains a byte[] containing the unique identifier for the Traffic Sensor System that was removed.

If the discriminator is CurrentStatus the union contains an array of one or more TSSStatus objects.

If the discriminator is ModeChanged, the union contains a ModeChangedEventInfo.

If the discriminator is OpStatusChanged, the union contains an OpStatusChangedEventInfo object.

5.18.1.42 TSSEventType (Class)

This enumeration defines the types of events that may be pushed on an event channel by a Transportation Sensor Status object. The values in this enumeration are used as the discriminator in the TSSEvent union.

ObjectAdded - a TransportationSensorSystem has been added to the system.

ObjectRemoved - a TransportationSensorSystem has been removed from the system.

CurrentStatus - The event contains the current status of one or more Transportation Sensor System objects.

ConfigChanged - One or more configuration values for the Transportation Sensor System have been changed.

ModeChanged - The communications mode of the TransportationSensorSystem has changed.

OpStatusChanged - The operational status of the TransportationSensorSystem has changed.

5.18.1.43 TSSListEntry (Class)

This struct is used to pass a TransportationSensorSystem object together with its ID. This struct is provided for convenience because when discovering an object, it is usually required to make a call to the object's getID() method.

5.18.1.44 TSSStatus (Class)

This class holds current status information for a TSS as follows:

m_id - The ID of the TSS for which this status applies.

m_zoneGrpTrafficParms - The traffic parameters for each ZoneGroup of the Transportation Sensor System as specified in the Sensor system's TSSConfiguration object.

m_mode - The communication mode of the TSS.

m_opStatus - The operational status for the TSS.

m_trafficParameterTimestamp - A timestamp that records when the traffic parameter data was collected from the device.

m_avgSpeed - average speed at the detector level.

m_speedRange - speed range at the detector level (avg speed).

5.18.1.45 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.18.1.46 ZoneGroup (Class)

This class is used to group one or more detection zones of a Transportation Sensor System into a logical grouping. Traffic parameters for all detection zones included in the group are averaged to provide a single set of traffic parameters for the group.

5.18.1.47 ZoneGroupTrafficParms (Class)

This struct contains traffic parameters for a ZoneGroup.

m_zoneGroupNumber - The number of the zone group for which the traffic parameters apply.

m_trafficParms - The traffic parameter values for the zone group.

m_zoneTrafficParms - zone parms for each zone in the group.

5.18.1.48 ZoneTrafficParms (Class)

This struct contains traffic parameters for a Zone.

m_zoneNumber - The number of the zone for which the traffic parameters apply.

m_trafficParms - The traffic parameter values for the zone.

5.18.1.49 TrafficEventManager (Class Diagram)

This class diagram contains all classes relating to Traffic Events

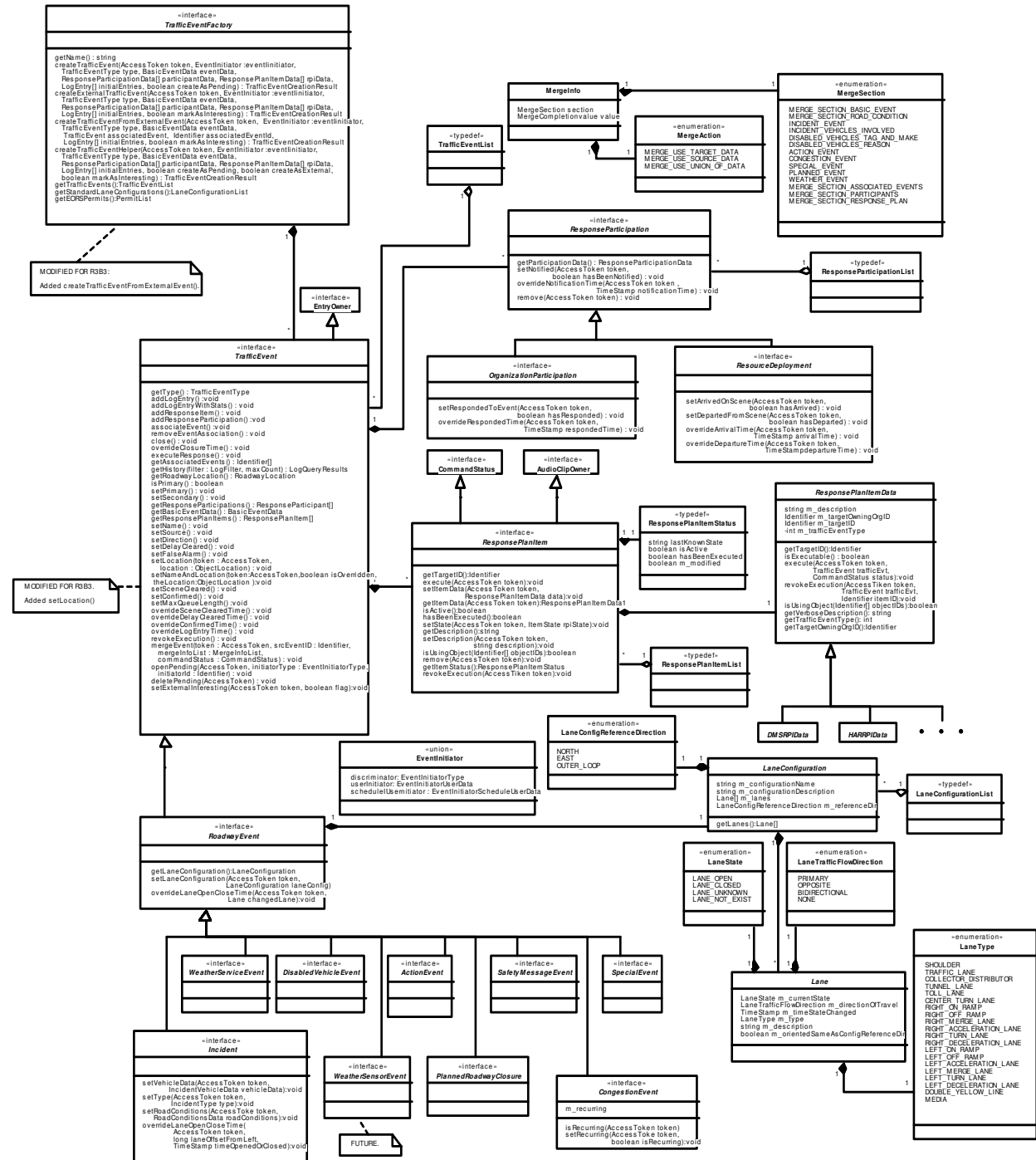


Figure 5-233. TrafficEventManager (Class Diagram)

5.18.1.49.1 ActionEvent (Class)

This class models roadway events that require an operations center to take action but do not fit well into the other event categories. An example of this type of event would be debris in the roadway.

5.18.1.49.2 AudioClipOwner (Class)

This interface allows the AudioClipManager to check whether there are any parties interested in an audio clip. If no AudioClipOwners claim interest in a clip, the clip can be deleted.

5.18.1.49.3 CommandStatus (Class)

The CommandStatus CORBA interface is used to allow a calling process to be notified of the progress of a long-running asynchronous operation. This is normally used when field communications are involved to complete a method call. The most common use is to allow a GUI to show the user the progress of an operation. It can also be used and watched by a server process when it needs to call on another server process to complete an operation. The long running operation typically calls back to the CommandStatus object periodically as the command is being executed, to provide in-progress status information, and it always makes a final call to the CommandStatus when the operation has completed. The final call to the CommandStatus from the long running operation indicates the success or failure of the command.

5.18.1.49.4 CongestionEvent (Class)

This class models roadway congestion which may be tagged as recurring or non-recurring through the use of an attribute.

5.18.1.49.5 DisabledVehicleEvent (Class)

This class models disabled vehicles on the roadway.

5.18.1.49.6 DMSRPIData (Class)

The DMSRPIData class is an abstract class which describes a response plan item for a DMS. It contains the unique identifier of the DMS to contain the DMSMessage, and the DMSMessage itself.

5.18.1.49.7 EntryOwner (Class)

Interface which must be implemented by any class which is responsible for putting an ArbQueueEntry on a device's arbitration queue. This validate method of this interface can be called by the device to determine continued validity of the entry (either during recovery or as a final check of the validity of an entry before putting its message on the device).

5.18.1.49.8EventInitiator (Class)

This union contains information about the entity or entities involved in the initiation of a traffic event. This can be the schedule, if a schedule was involved in initiating the event, and/or a user, if a user was involved in initiating the event. This union allows for possible expansion in future releases, where traffic events may be initiated by a schedule without user confirmation, or by CHART devices (traffic sensors, weather sensors, etc.) or external interfaces (RITIS, etc.) initially with, or possibly later without, user involvement.

5.18.1.49.9HARRPIData (Class)

This class represents an item in a traffic event response plan that is capable of issuing a command to put a message on a HAR when executed. When the item is executed, it adds an ArbQueueEntry to the specified HAR, which stores the entry in its MessageQueue. When the item's execution is revoked, or the item is removed from the response plan (manually or implicitly through closing the traffic event) the item asks the HAR to remove the entry. The HARRPIData object also allows specification of a subset (0 to all) of the HARNotifier devices (SHAZAM or DMS devices acting as SHAZAMs) to be activated if and while the message is being broadcast on the HAR.

5.18.1.49.10 Incident (Class)

This class models objects representing roadway incidents. An incident typically involves one or more vehicles and roadway lane closures.

5.18.1.49.11 Lane (Class)

This class represents a single traffic lane at the scene of a RoadwayEvent.

5.18.1.49.12 LaneConfigReferenceDirection (Class)

This enumeration restricts the possible reference directions for a lane configuration, which is necessary because the lane offsets are defined relative to the "left" side, which is an ambiguous term. For example, if the direction is North then "left" to the West, but if the direction is South (also valid on a North-South roadway) then "left" could be considered (if not for this enumeration) to East. Thus if the direction of the lane config were to change from North to South, the lanes would "flip" unintentionally. This enumeration holds the reference direction for a North-South roadway to always be to the West (regardless of whether the direction of the event is North or South), and holds similarly for East-West roadways and beltways (Inner-Outer loops).

5.18.1.49.13 LaneConfiguration (Class)

This class contains data that represents the configuration of the lanes.

5.18.1.49.14 LaneConfigurationList (Class)

A collection of LaneConfiguration objects.

5.18.1.49.15 LaneState (Class)

This enumeration lists the possible states that a traffic lane may be in.

5.18.1.49.16 LaneTrafficFlowDirection (Class)

Defines the possible directions of traffic flow, relative to the lane orientation.

5.18.1.49.17 LaneType (Class)

This enumeration lists the types of lanes.

5.18.1.49.18 MergeAction (Class)

This enumeration specifies how to merge a section of data during a traffic event merge operation.

5.18.1.49.19 MergeInfo (Class)

This valuetype is passed between Chartlite to Chart to provide instructions for performing the merge

5.18.1.49.20 MergeSection (Class)

This idl enum defines values for each merge section

5.18.1.49.21 OrganizationParticipation (Class)

This class is used to manage the data captured when an operator notifies another organization of a traffic event.

5.18.1.49.22 PlannedRoadwayClosure (Class)

This class models planned roadway closures such as road construction. This interface will be expanded in future releases to include interfacing with the EORS system.

5.18.1.49.23 ResourceDeployment (Class)

This class is used to store the data captured when an operator deploys resources to the scene of a traffic event.

5.18.1.49.24 ResponseParticipation (Class)

This interface represents the involvement of one particular resource or organization in response to a particular traffic event.

5.18.1.49.25 ResponseParticipationList (Class)

A collection of ResponseParticipation objects.

5.18.1.49.26 ResponsePlanItem (Class)

Objects of this type can be executed as part of a traffic event response plan. A ResponsePlanItem can be executed by an operator, at which time it becomes the responsibility of the System to activate the item on the ResponseDevice as soon as it is appropriate.

5.18.1.49.27 ResponsePlanItemData (Class)

This class is a delegate used to perform the execute and remove tasks for the response plan item. Derived classes of this base class have specific implementations for the type of device the response plan item is used to control.

5.18.1.49.28 ResponsePlanItemList (Class)

A collection of ResponsePlanItem objects.

5.18.1.49.29 ResponsePlanItemStatus (Class)

This structure contains data that describes the current state of a response plan item.

5.18.1.49.30 RoadwayEvent (Class)

This class models any type of incident that can occur on a roadway. This point in the hierarchy provides a break off point for traffic event types that pertain to other modals.

5.18.1.49.31 SafetyMessageEvent (Class)

This type of event is created by an operator when he/she would like to send a safety message to a device.

5.18.1.49.32 SpecialEvent (Class)

This class models special events that affect roadway conditions such as a concert or professional sporting event.

5.18.1.49.33 TrafficEvent (Class)

Objects of this type represent traffic events that require action from system operators.

5.18.1.49.34 TrafficEventFactory (Class)

This interface is supported by objects that are capable of creating traffic event objects in the system.

5.18.1.49.35 TrafficEventList (Class)

A collection of TrafficEvent objects.

5.18.1.49.36 WeatherSensorEvent (Class)

This class models roadway weather events such as snow or fog that are reported by the system's weather monitoring devices. Operators will need to manually enter the information in these events for this release. In future releases, these events will be automatically generated by the system.

5.18.1.49.37 WeatherServiceEvent (Class)

This class models roadway weather events such as snow or fog that are manually entered by an operator in response to receiving an alert from the national weather service.

5.18.1.50 TrafficEventManager2 (Class Diagram)

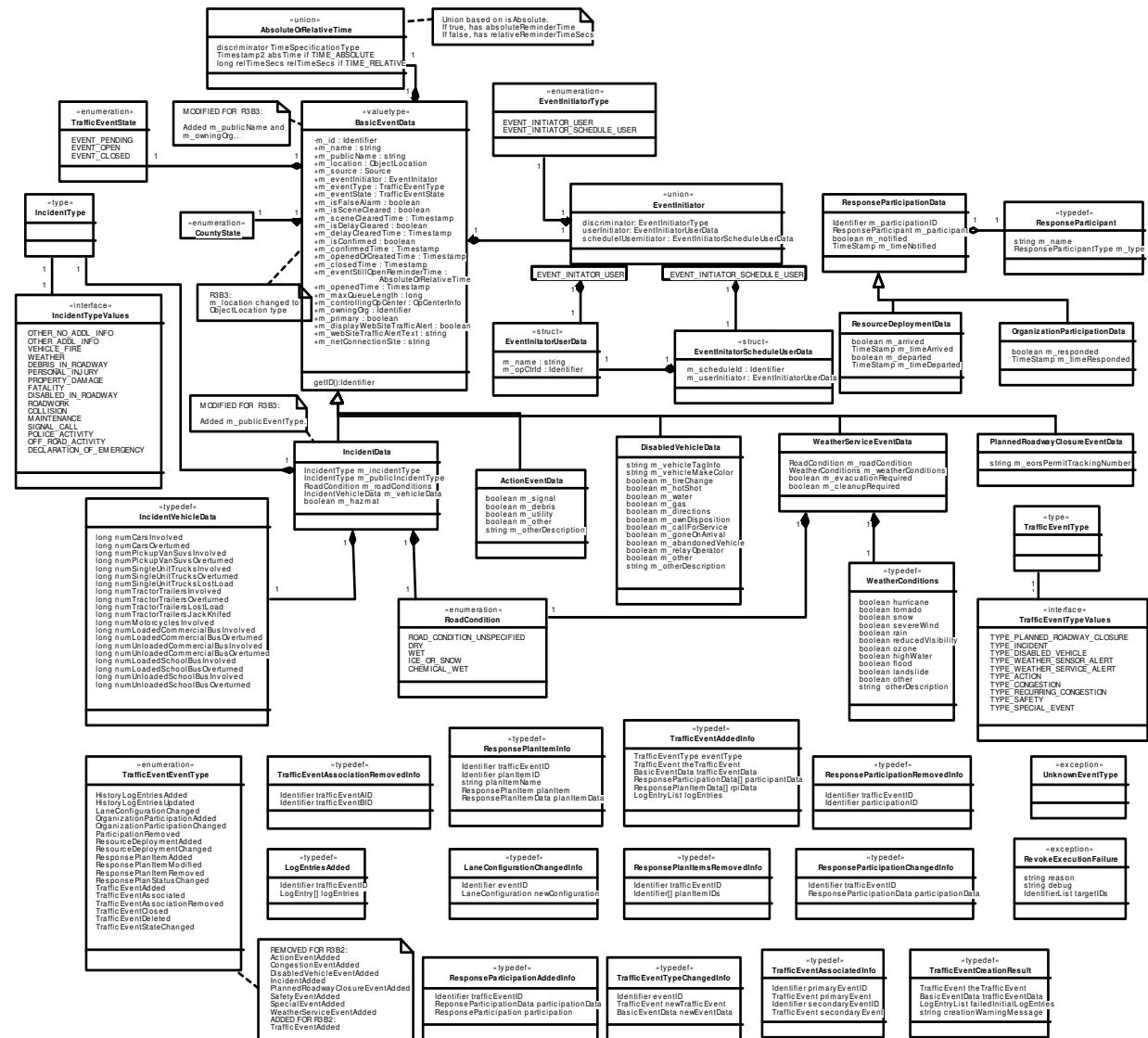


Figure 5-234. TrafficEventManager2 (Class Diagram)

5.18.1.50.1 AbsoluteOrRelativeTime (Class)

This union stores a time, in either absolute or relative terms.

5.18.1.50.2ActionEventData (Class)

This class represents all data specific to an Action event type traffic event.

5.18.1.50.3BasicEventData (Class)

This class represents the data common to all traffic events. All derived data types will inherit all data shown in this class.

5.18.1.50.4CountyState (Class)

This enumeration defines the various counties in Maryland and the states surrounding Maryland that will be used for defining the traffic event.

5.18.1.50.5DisabledVehicleData (Class)

This class represents all data specific to a disabled vehicle traffic event.

5.18.1.50.6EventInitiatorScheduleUserData (Class)

This structure contains data about a schedule involved in the initiation of a traffic event. It is contained within the EventInitiator union.

5.18.1.50.7EventInitiatorUserData (Class)

This structure contains data about a user involved in the initiation of a traffic event. It is contained within the EventInitiator union.

5.18.1.50.8EventInitiator (Class)

This union contains information about the entity or entities involved in the initiation of a traffic event. This can be the schedule, if a schedule was involved in initiating the event, and/or a user, if a user was involved in initiating the event. This union allows for possible expansion in future releases, where traffic events may be initiated by a schedule without user confirmation, or by CHART devices (traffic sensors, weather sensors, etc.) or external interfaces (RITIS, etc.) initially with, or possibly later without, user involvement.

5.18.1.50.9EventInitiatorType (Class)

This enumeration identifies the types of initiators which can initiate traffic events. Traffic events can be initiated by a user (directly), or by a schedule (with user involvement). This enumeration, and the union in which it is a discriminator, allows for possible expansion in future releases, where traffic events may be initiated by a schedule without user confirmation, or by CHART devices (traffic sensors, weather sensors, etc.) or external interfaces (RITIS, etc.) initially with, or possibly later without, user involvement.

5.18.1.50.10 IncidentData (Class)

This class represents data specific to an Incident type traffic event.

5.18.1.50.11 IncidentType (Class)

This typedef defines the type of the incident.

5.18.1.50.12 IncidentTypeValues (Class)

This interface lists all possible incident types.

5.18.1.50.13 IncidentVehicleData (Class)

This class represents the vehicles involved data for incidents. Its purpose is to simplify the exchange of data between GUI and server.

5.18.1.50.14 LaneConfigurationChangedInfo (Class)

This structure contains the data that is broadcast when the lane configuration of a traffic event is changed.

5.18.1.50.15 LogEntriesAdded (Class)

This structure contains the data that is broadcast when new entries are added to the event history log of a traffic event.

5.18.1.50.16 OrganizationParticipationData (Class)

This class represents the data required to describe an organization's participation in the response to a traffic event.

5.18.1.50.17 PlannedRoadwayClosureEventData (Class)

This class contains data specific to the PlannedRoadwayEvent type of traffic event.

5.18.1.50.18 ResourceDeploymentData (Class)

This class represents the data required to describe a resource's participation in the response to a traffic event.

5.18.1.50.19 ResponseParticipant (Class)

The ResponseParticipant class is a non-behavioral structure which specifies a participant in a response.

5.18.1.50.20 ResponseParticipationAddedInfo (Class)

This structure contains the data that is broadcast when a response participant is added to the response to a particular traffic event.

5.18.1.50.21 ResponseParticipationChangedInfo (Class)

This structure contains the data pushed in a CORBA event any time any type of response participation object changes state.

5.18.1.50.22 ResponseParticipationData (Class)

This class contains all data pertinent to any class that represents a response participation.

5.18.1.50.23 ResponseParticipationRemovedInfo (Class)

This structure contains the data that is broadcast when one or more response plan items are removed from a traffic event.

5.18.1.50.24 ResponsePlanItemInfo (Class)

This structure contains the data that is broadcast any time a new response plan item is added or an existing response plan item is modified.

5.18.1.50.25 ResponsePlanItemsRemovedInfo (Class)

This structure contains the data that is broadcast when one or more response plan items are removed from a traffic event.

5.18.1.50.26 RevokeExecutionFailure (Class)

This class defines a exception thrown when failed to revoke a response plan item's execution.

5.18.1.50.27 RoadCondition (Class)

This enumeration lists the possible roadway conditions at the scene of a traffic event.

5.18.1.50.28 TrafficEventAddedInfo (Class)

This structure contains the data that is broadcast when a new traffic event is added to the system.

5.18.1.50.29 TrafficEventAssociatedInfo (Class)

This structure contains the data that is broadcast when two traffic events are associated.

5.18.1.50.30 TrafficEventAssociationRemovedInfo (Class)

This structure contains the data that is broadcast when the association between two traffic events is removed.

5.18.1.50.31 TrafficEventCreationResult (Class)

This result is returned from createEvent() to indicate warning messages if the event was not created cleanly.

5.18.1.50.32 TrafficEventEventType (Class)

his enumeration defines the types of CORBA events that can be broadcast on a Traffic Event related CORBA Event channel.

5.18.1.50.33 TrafficEventState (Class)

This enumeration lists the possible states for a traffic event. The states are pending, open,

and closed. A false alarmed "state" is considered a special case of "closed", so false alarmed events will have a TrafficEventState of EVENT_STATE_CLOSED. They will also have the m_isFalseAlarm flag in their BasicEventData set to true to distinguish them from normally closed events.

5.18.1.50.34 TrafficEventType (Class)

This typedef defines the type of traffic event.

5.18.1.50.35 TrafficEventTypeChangedInfo (Class)

This structure contains the data that is broadcast when a traffic event changes types. The traffic event object that represented the traffic event previously is removed from the system and is replaced by the newTrafficEvent reference contained in this structure. If the consumer of this CORBA event has stored any references to the traffic event previously, those references should be replaced with this new reference.

5.18.1.50.36 TrafficEventTypeValues (Class)

This interface defines the types of traffic events that are supported by the system.

5.18.1.50.37 UnknownEventType (Class)

This class defines a exception thrown when the type of a traffic event type is not known and is not defined in TrafficEventTypeValues.

5.18.1.50.38 WeatherConditions (Class)

This structure contains all possible weather conditions. Each member should be set to true if that condition applies, false otherwise. The m_otherDescription member will only be considered valid if the m_other member is set to true.

5.18.1.50.39 WeatherServiceEventData (Class)

This class contains data specific to the WeatherServiceEvent type of traffic event.

5.18.1.51 TrafficEventRule (Class Diagram)

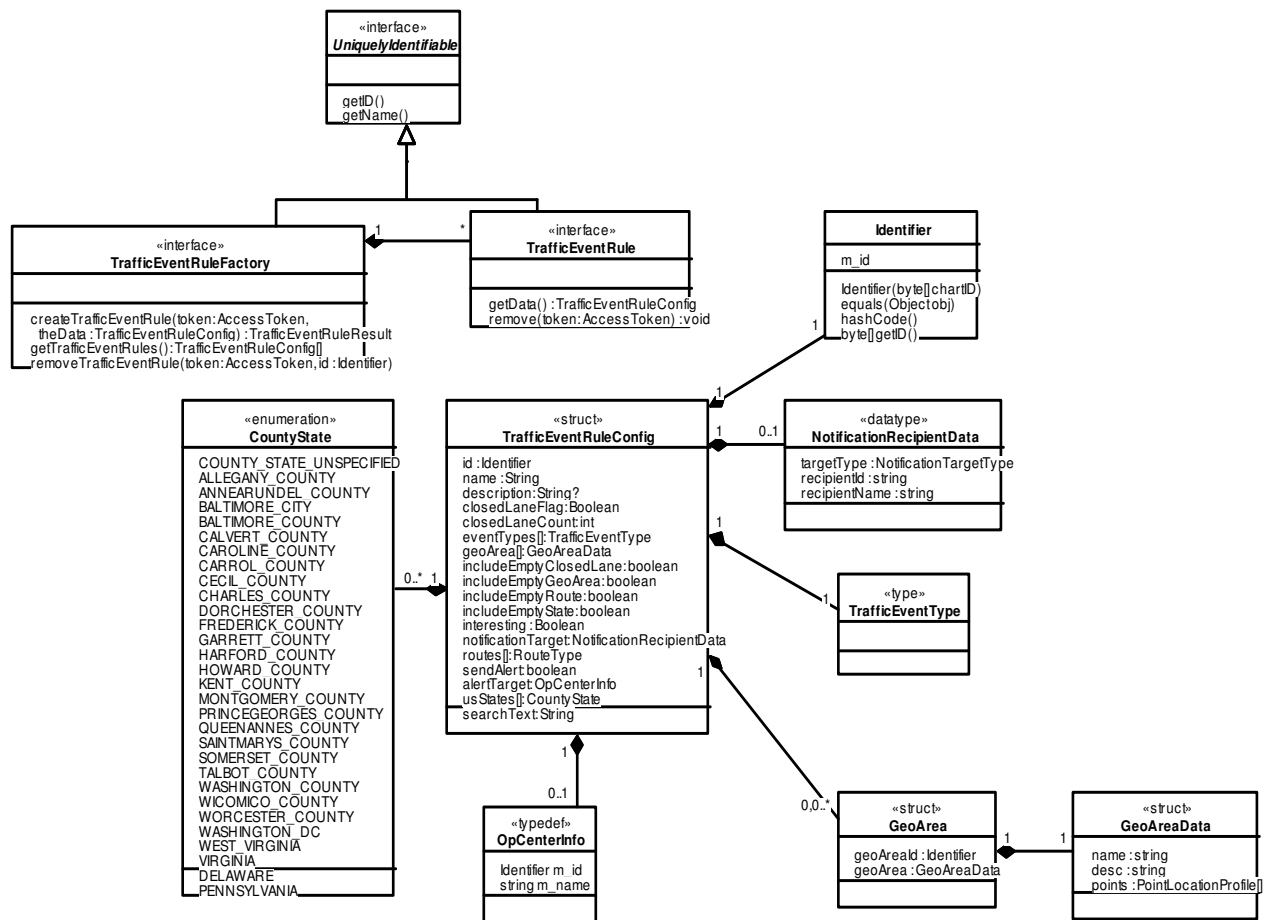


Figure 5-235. TrafficEventRule (Class Diagram)

5.18.1.51.1 CountyState (Class)

This enumeration defines the various counties in Maryland and the states surrounding Maryland that will be used for defining the traffic event.

5.18.1.51.2 GeoArea (Class)

The GeoArea struct defines a unique GeoArea within the CHART system. It has a unique id and a GeoAreaData struct.

5.18.1.51.3 GeoAreaData (Class)

The GeoArea struct is a simple representation of a polygon (ordered list of points) defining a Geographical Area within the CHART system.

5.18.1.51.4 Identifier (Class)

Wrapper class for a CHART2 identifier byte sequence. This class will be used to add

identifiable objects to hash tables and perform subsequent lookup operations.

5.18.1.51.5NotificationRecipientData (Class)

This object contains the data that is returned as a result of an object get recipient (groups or individuals) request..

5.18.1.51.6OpCenterInfo (Class)

This structure contains the information about an OperationsCenter.

5.18.1.51.7TrafficEventRule (Class)

5.18.1.51.8TrafficEventRuleConfig (Class)

5.18.1.51.9TrafficEventRuleFactory (Class)

5.18.1.51.10 TrafficEventType (Class)

This typedef defines the type of traffic event.

5.18.1.51.11 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

5.18.1.52 TravelRouteManagement (Class Diagram)

This Class Diagram shows the CORBA system interface classes and methods used to manage Roadway Links and Travel Routes, added in R3B3. Each travel route in the system can track travel times (accumulated from roadway links configured into the route) and/or toll rates. Link travel times and toll rates are acquired by separate CHART import processes which call in when data updates are available (every few minutes).

A factory serves roadway links read from the database and travel routes configured by CHART administrators. Roadway links are imported into the database as an offline process from a large dataset provided by from CHART's travel time provider(s), and are passed around as configuration and data (stats) structures (i.e., there is no RoadwayLink "interface"). Travel routes are created and configured on as necessary by CHART administrators, and each travel route is modeled as a TravelRoute interface which provides access to its own configuration and data directly.

can call when it has updated link travel time available to provide.

5.18.1.52.4 LinkRawData (Class)

This structure is used by the CHART roadway link travel time import service to pass raw link travel time data acquired by a third party travel time provider into the CHART TravelRoute system. Many of these is expected to be passed in, in an array, at one time. An update does not have to contain the entire collection of links tracked by CHART, but it is expected to be a large chunk of them (for example, all freeways in Maryland is one chunk). In R3B3, INRIX was added as a travel time provider for the CHART system.

5.18.1.52.5 LinkTravTimeHistRecord (Class)

This structure is used to store the most recent X historical link travel time data points acquired by the system. This is a circular array, with the head (oldest record) referenced in the LinkTravelTimeHistStats. The tail (newest record) is head-1 (mod X), and is always the same data point as is contained in the LinkTravelTimeStats structure.

5.18.1.52.6 LinkTravTimeHistStats (Class)

This structure (together with the LinkTravelTimeHistRecord) stores the most recent X historical link travel time data points. It contains the head pointer and the LinkTravelTimeHistRecord array which contains the actual data. This structure is not pushed on the event channel, because it normally contains data which clients could have already accumulated themselves, but is available on demand.

5.18.1.52.7 LinkTravTimeStats (Class)

This structure contains the most recent travel time data point acquired for a roadway link. It matches the most recent record in the LinkTravelTimeHistStats.

5.18.1.52.8 RoadwayLinkConfig (Class)

This structure contains the configuration data for a roadway link. It includes the external system name (e.g., "INRIX"), the ID by which the external system identifies the link, and location data.

5.18.1.52.9 RoadwayLinkConfigInfo (Class)

This is a convenience structure which combines a roadway link ID with the RoadwayLinkConfig. It is used for passing configuration data about all links in the system or all links in one route from the Travel Route Module to clients.

5.18.1.52.10 RoadwayLinkFullInfo (Class)

This is a convenience structure which combines a roadway link ID with its configuration, current, and historical stats. It is used for passing the full set of data for all links in the system or all links in one route from the Travel Route Module to clients.

5.18.1.52.11 RouteAndLinkConfig (Class)

This convenience structure is used to pass configuration data for a travel route plus configuration data for all the route's links from the Travel Route Module to clients.

5.18.1.52.12 RouteConfigEvent (Class)

This structure is contained in a TravelRouteEvent CORBA event union which is pushed when a route is added or a route configuration is updated.

5.18.1.52.13 RouteFullStats (Class)

This convenience structure holds all stats data for a single travel route -- current stats and recent historical stats for the route and for the constituent links (if any).

5.18.1.52.14 RouteHistStats (Class)

This structure is used to store historical statistical data (travel times and toll rates) for a Travel Route. It consists of an array of zero or one RouteTravelTimeHistStats, storing Travel Time historical statistics (if configured with links), and an array of zero or one RouteTollRateHistStats, storing Toll Rate historical statistics (if configured to track toll rates).

5.18.1.52.15 RouteLink (Class)

This structure makes the association between a travel route and one roadway link which helps comprise the route, together with parameters associated with the use of the link within that particular route: the percent of the link to include in the route, and the minimum acceptable quality for link travel time data as used in that particular route.

5.18.1.52.16 RouteLinkHistStats (Class)

This convenience structure contains historical travel time stats for a travel route and for all the links which comprise the route. It is available on demand from TravelRoute.

5.18.1.52.17 RouteLinkStats (Class)

This convenience structure contains current travel time stats for a travel route and for all the links which comprise the route. It is available on demand from TravelRoute.

5.18.1.52.18 RouteStats (Class)

This convenience structure combines the current travel time data and toll rate data for a travel route. (It contains zero or one of each, depending on what types of data the route is configured to track.) It is available on demand via the TravelRoute.

5.18.1.52.19 RouteTollRateHistRecord (Class)

This structure is used to store the most recent X historical route toll rate data points accumulated by the system. This is a circular array, with the head (oldest record)

referenced in the RouteTollRateHistStats. The tail (newest record) is head-1 (mod X), and is always the same data point as is contained in the RouteTollRateStats structure.

5.18.1.52.20 RouteTollRateHistStats (Class)

This structure (together with the RouteTollRateHistRecord) stores the most recent X historical route toll rate data points. It contains the head pointer and the RouteTollRateHistRecord array which contains the actual data. This structure is not pushed on the event channel, because it normally contains data which clients could have already accumulated themselves, but is available on demand.

5.18.1.52.21 RouteTollRateStats (Class)

This structure contains the current toll rate data for a travel route. This includes the time the rate became effective and the toll rate itself. This data is also provided in the most recent entry in the history structure. The toll rate field may contain a negative number defined by StatsConstants, which indicates an error. There are two other fields NOT provided in the history structure -- the time the toll rate expires, and a reason string. This will be the empty string if the toll rate has been successfully provided recently, or details on the error condition if an error constant is specified.

5.18.1.52.22 RouteTollRateUpdate (Class)

This structure is contained in a TravelRouteEvent for toll rate updates. It contains the ID of the route being updated and the RouteTollRateStats containing the new toll rate data (only the current toll rate data is provided in the event, to reduce the CORBA event size. GUIs are expected to cache toll rate data and build up their own history data (although history data is available on demand from the TravelRoute).

5.18.1.52.23 RouteTollRateUpdateEvent (Class)

This structure is one of several which can be the element of a TravelRouteEvent union. It contains the ID of the Travel Route to which the toll rate statistics apply, and the toll rate data itself.

5.18.1.52.24 RouteTravTimeHistRecord (Class)

This structure is used to store the most recent X historical route travel time data points accumulated by the system. This is a circular array, with the head (oldest record) referenced in the RouteTravelTimeHistStats. The tail (newest record) is head-1 (mod X), and is always the same data point as is contained in the RouteTravelTimeStats structure.

5.18.1.52.25 RouteTravTimeHistStats (Class)

This structure (together with the RouteTravelTimeHistRecord) stores the most recent X historical route travel time data points. It contains the head pointer and the RouteTravelTimeHistRecord array which contains the actual data. This structure is not pushed on the event channel, because it normally contains data which clients could have

already accumulated themselves, but is available on demand.

5.18.1.52.26 RouteTravTimeStats (Class)

This structure contains the current travel time data for a travel route. This includes the time the travel time was computed, and the computed speed. This data is also provided in the most recent entry in the history structure. The travel time may contain a negative number defined by StatsConstants, which indicates an error. There are two other fields NOT provided in the history structure -- a computed trend (UP, DOWN, or FLAT) and a reason string. This will be the travel time calculation if it has been computed successfully, or details on the error condition if an error constant is specified.

5.18.1.52.27 RouteTravTimeUpdate (Class)

This structure is contained in a TravelRouteEvent for travel time updates. It contains the ID of the route being updated and the RouteTravelTimeStats containing the new travel time data and the RouteLinkStats containing the new travel times for all the links in the route. (Only the current travel time data for the route and links is provided in the event, to reduce the CORBA event size. GUIs are expected to cache travel time data and build up their own history data -- although history data is available on demand from the TravelRoute.)

5.18.1.52.28 RouteTravTimeUpdateEvent (Class)

This structure is one of several which can be the element of a TravelRouteEvent union. It contains the ID of the Travel Route to which the travel time data applies, and the travel time data itself.

5.18.1.52.29 StatsState (Class)

This interface defines constants that can be used to populate the various "Stats" objects: routeTollRateCents in RouteTollRateStats and RouteTollRateHistRecord; and routeTvlTimeSecs in RouteTravelTimeStats and RouteTravelTimeHistRecord. These constants are not used in link-level stats objects.

5.18.1.52.30 TollDataConsumer (Class)

CORBA interface that must be implemented by any CHART component that wants to be notified when toll rates changes.

5.18.1.52.31 TollRateConfig (Class)

This structure holds the part of the Travel Route configuration pertaining to toll rates, if the travel route is configured to track toll rates. This structure holds the part of the Travel Route configuration pertaining to toll rates. One of these is contained in the TravelRouteConfig if the travel route is configured to track toll rates, otherwise there is none.

5.18.1.52.32 TollRateRouteInfo (Class)

This structure is used to pass information on toll rate routes for which we are currently receiving toll rates. Only toll rate routes which were included in the most recent post of toll rate data are maintained at any given time.

5.18.1.52.33 TollRawData (Class)

This structure is used by the CHART importer of toll rates to pass toll rate data into the travel route factory. The intent is that the importer will receive all toll rates in one transaction and will pass all toll rates on to the TravelRouteFactory in one transaction. In R3B3, Vector was added as a toll rate provider for the CHART system.

5.18.1.52.34 TravelRoute (Class)

This is the primary CORBA interface for working with travel routes in CHART. This interface provides methods for getting various collections of configuration and/or statistical data for a travel route. It also provides methods for objects to register to be TravelRouteConsumer for the travel route (for instances, DMSs that have the route enabled in a traveler information message). Finally it provides methods for updating and removing travel routes.

5.18.1.52.35 TravelRouteConfig (Class)

This structure holds the part of the Travel Route configuration pertaining to travel times, if the travel route is configured to track travel times. It contains the IDs of the links comprising the route, but not the link configurations themselves. (See RouteAndLinkConfig.)

5.18.1.52.36 TravelRouteConsumer (Class)

This interface allows other CHART objects to register as a direct consumer of travel route statistical data. It provides operations for the travel route to call when the travel time or toll rate for the route is updated. A DMS registers as a TravelRouteConsumer when a TravelerInfoMsg is enabled.

5.18.1.52.37 TravelRouteConsumerInfo (Class)

This convenience structure lists a TravelRouteConsumer ID, reference, and type.

5.18.1.52.38 TravelRouteConsumerType (Class)

This enumeration lists the types of CHART objects which can be a TravelRouteConsumer. Starting in R3B3, the CHART2DMS is configured to implement the TravelRouteConsumer interface.

5.18.1.52.39 TravelRouteDisplayConfig (Class)

5.18.1.52.40 TravelRouteEvent (Class)

This union defines the various types of CORBA events pushed on the Travel Route CORBA event channel, and the content of the events for each type of events. Events are configured for routes being added, reconfigured, and deleted, and for updates to route travel time data and toll rate data. Note that there are no link management events, as links cannot be added, modified, or removed. Link data updates are included in the RouteTravelTimeUpdateEvent for each route which they are a part of.

5.18.1.52.41 TravelRouteEventType (Class)

This enumeration lists the various types of events pushed on the Travel Route CORBA event channel. This enumeration is the discriminator for the TravelRouteEvent union pushed in every CORBA event.

5.18.1.52.42 TravelRouteFactory (Class)

This interface is the entry point for the Travel Route Management. It serves up travel routes (also an interface) and roadway links (structure data). It provides various operations for acquiring travel routes and roadway links. Since roadway links are not maintained as a separate interface, the factory provides the primary operations for acquiring link configuration and statistical data (although a TravelRoute interface also provides methods for acquiring such data about the links directly associated with it).

5.18.1.52.43 TravelRouteInfo (Class)

This convenience structure contains the ID of and a reference to a TravelRoute. It is used by the factory when informing clients of travel routes.

5.18.1.52.44 TravTimeConfig (Class)

This structure holds the part of the Travel Route configuration pertaining to travel times. One of these is contained in the TravelRouteConfig if the travel route is configured to track travel times, otherwise there is none.

5.18.1.52.45 TravTimeQuality (Class)

This enumeration has been replaced with a simple integer indication of quality for a link travel time. The raw integer travel time quality indicator provided by the provider (e.g., INRIX currently uses 30, 20, 10), will be used directly (untranslated) to populate the travel time quality indicator in the CHART classes which were originally designed with a TravTimeQuality enumeration. This will work for any travel time provider who provides an integer quality indicator where higher numbers indicate greater quality. (There have recently been suggestions intimated that INRIX may modify their quality indicator scheme, to 10, 20, 25, 30; 1-5; 1-10; or possibly even 1-100.)

5.18.1.52.46 TravTimeTrend (Class)

This enumeration lists the possible values for a travel time trend. A travel route's trend can be either UP, FLAT, or DOWN, or the trend can be undefined (for instance if the link quality threshold has not been met).

5.18.1.52.47 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness..

5.18.1.53 TravelRouteManagement2 (Class Diagram)

This diagram shows some toll rate specific classes related to travel routes.

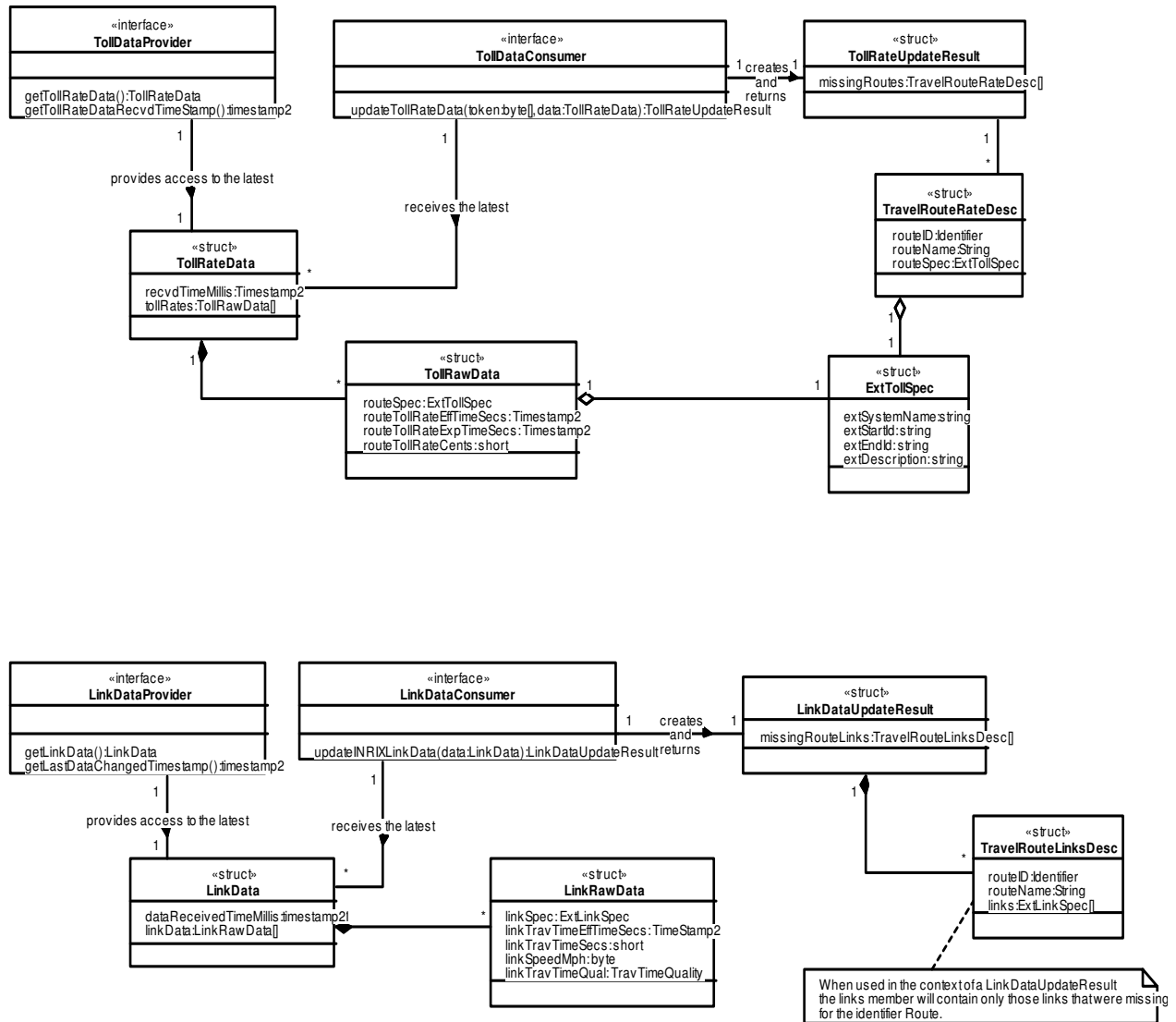


Figure 5-237. TravelRouteManagement2 (Class Diagram)

5.18.1.53.1 ExtTollSpec (Class)

This structure is used to identify a toll rate route. It contains the supplying external system, the start ID and end ID of the toll rate route (which is the "key" used to identify the toll rate route, and the name by which the external system refers to the route.

5.18.1.53.2LinkData (Class)

This class contains the latest LinkRawData along with a timestamp that indicates when it was obtained by the CHART system.

5.18.1.53.3LinkDataConsumer (Class)

This CORBA interface defines the methods that a consumer of INRIX link data must implement in order to be updated when data changes.

5.18.1.53.4LinkDataProvider (Class)

This CORBA interface defines the methods that a provider of INRIX link data must implement.

5.18.1.53.5LinkDataUpdateResult (Class)

This class is used to return results from a consumer link data update.

5.18.1.53.6LinkRawData (Class)

This structure is used by the CHART roadway link travel time import service to pass raw link travel time data acquired by a third party travel time provider into the CHART TravelRoute system. Many of these is expected to be passed in, in an array, at one time. An update does not have to contain the entire collection of links tracked by CHART, but it is expected to be a large chunk of them (for example, all freeways in Maryland is one chunk). In R3B3, INRIX was added as a travel time provider for the CHART system.

5.18.1.53.7TollDataConsumer (Class)

CORBA interface that must be implemented by any CHART component that wants to be notified when toll rates changes.

5.18.1.53.8TollDataProvider (Class)

System interface that is implemented by services that provide toll rate data to the system.

5.18.1.53.9TollRateData (Class)

This struct provides a collection of TollRawData objects along with a timestamp of when they were posted by the toll rate source.

5.18.1.53.10 TollRateUpdateResult (Class)

This class is used to return results from a call to a TollDataConsumer to update toll rates.

5.18.1.53.11 TollRawData (Class)

This structure is used by the CHART importer of toll rates to pass toll rate data into the travel route factory. The intent is that the importer will receive all toll rates in one

transaction and will pass all toll rates on to the TravelRouteFactory in one transaction. In R3B3, Vector was added as a toll rate provider for the CHART system.

5.18.1.53.12 TravelRouteLinksDesc (Class)

This struct provides the id and name of a route and alist of links that are associated with that route.

5.18.1.53.13 TravelRouteRateDesc (Class)

This class describes the toll rate that is currently configured for a CHART Travel route.

5.18.1.54 UserManagement (Class Diagram)

This class diagram contains the interfaces necessary to manage and utilize user profiles.

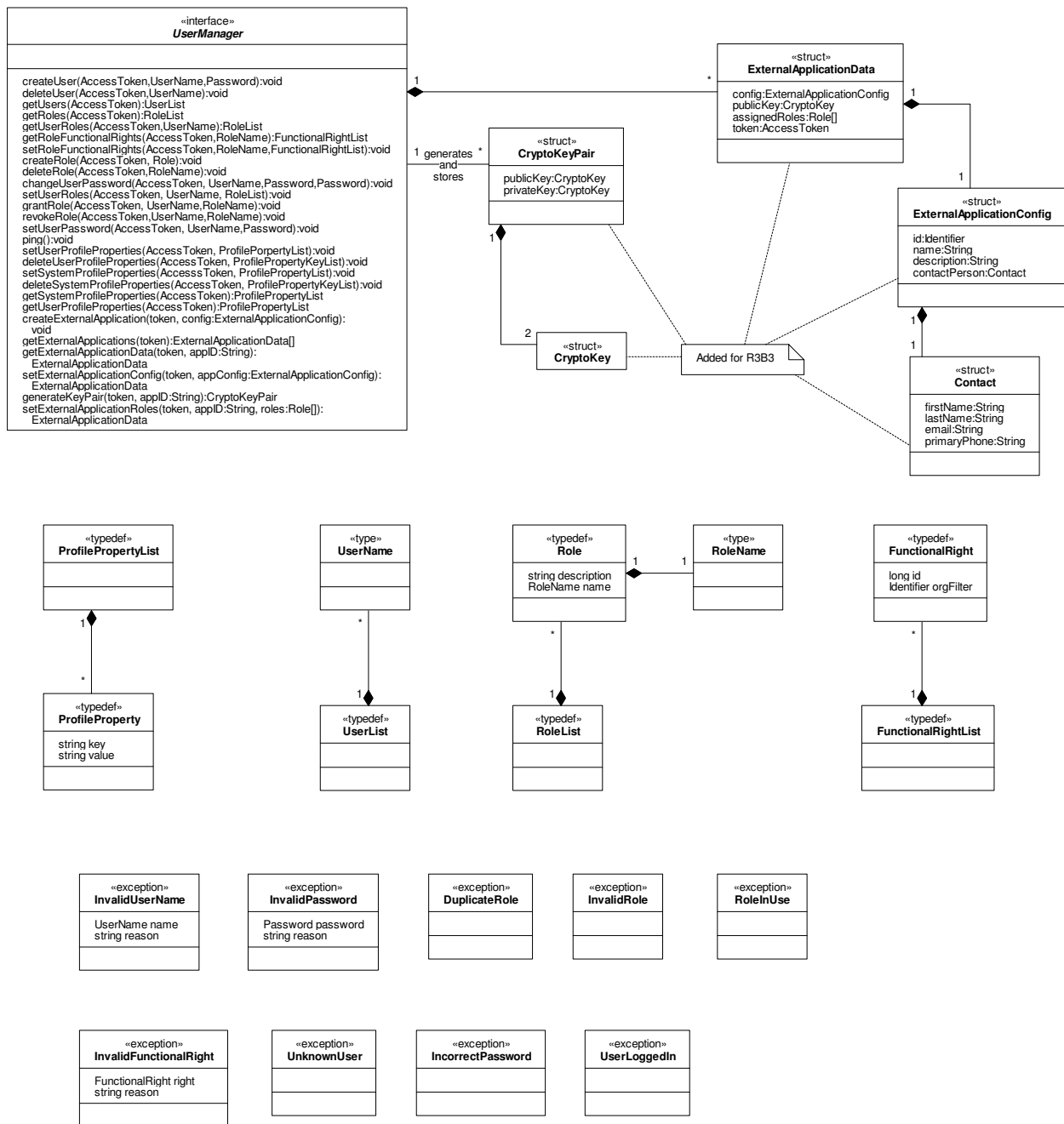


Figure 5-238. UserManagement (Class Diagram)

5.18.1.54.1 Contact (Class)

This class defines basic Contact data.

5.18.1.54.2 CryptoKey (Class)

This class represents a single key in a public/private key pair. It is an abstraction for an

array of bytes.

5.18.1.54.3CryptoKeyPair (Class)

This class represents a public/private key pair used by external client applications when communicating with CHART.

5.18.1.54.4DuplicateRole (Class)

This class represents an exception thrown when an attempt is made to define a role which already exists.

5.18.1.54.5ExternalApplicationConfig (Class)

This class holds configuration data for an external application.

5.18.1.54.6ExternalApplicationData (Class)

This class defines the data available for an external application.

5.18.1.54.7FunctionalRight (Class)

A functional right represents a particular user capability. A functional right grants a particular capability to perform system functions. Each functional right may be limited by attaching the identifier of a particular organization to which this right is constrained. This capability allows an administrator to grant a particular Role the ability to modify only shared resources owned by the identified organization. The orgFilter identifier CHART2 will allow access to any organizations shared resources.

5.18.1.54.8FunctionalRightList (Class)

A list of functional rights.

5.18.1.54.9IncorrectPassword (Class)

This class represents an exception thrown when the password specified for a user does not match that user's password in the database.

5.18.1.54.10 InvalidFunctionalRight (Class)

This class represents an exception thrown when an attempt is made to add an invalid functional right to a role.

5.18.1.54.11 InvalidPassword (Class)

This class represents an exception thrown when the password specified is invalid.

5.18.1.54.12 InvalidRole (Class)

This class represents the exception thrown when the specified role name does not exist in

the database.

5.18.1.54.13 InvalidUserName (Class)

This class represents an exception thrown when the username specified is not valid.

5.18.1.54.14 ProfileProperty (Class)

This class represents a key value pair that can be used to store system properties in the system database.

5.18.1.54.15 ProfilePropertyList (Class)

A list of profile properties.

5.18.1.54.16 Role (Class)

A Role is a collection of functional rights. A Role can be granted to a user, thus granting the user all functional rights contained within the role.

5.18.1.54.17 RoleInUse (Class)

This class represents an exception thrown when an attempt is made to delete a role which has users assigned to it.

5.18.1.54.18 RoleList (Class)

This structure contains a list of roles.

5.18.1.54.19 RoleName (Class)

Name assigned to a role. The role name must be unique and must be no longer than 32 bytes.

5.18.1.54.20 UnknownUser (Class)

This class represents an exception thrown when a user name is passed that is not in the user database.

5.18.1.54.21 UserList (Class)

A list of user names.

5.18.1.54.22 UserLoggedIn (Class)

This class represents an exception thrown when an attempt is made to delete a user who is currently logged in.

5.18.1.54.23 UserManager (Class)

The UserManager provides access to data dealing with user management. This includes

users, roles, and functional rights. The UserManager is largely an interface to the User Management database tables.

5.18.1.54.24 UserName (Class)

This typedef defines the type of UserName fields used in system interfaces.

5.19.1.1.1 CommFailureDB (Class)

This class is a utility used to log an entry in the Comm Failure log table in the database. This table is used to log details about any comm failure that occurs in the system.

5.19.1.1.2 java.util.Timer (Class)

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

5.19.1.1.3 java.util.TimerTask (Class)

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

5.19.1.1.4 LogFile (Class)

This class creates a flat file for writing system trace log messages and purges them at user specified interval. The log files created by this class are used for system debugging and maintenance only and are not to be confused with the system operations log which is modeled by the OperationsLog class.

5.19.1.1.5 ModemPortLocator (Class)

This class provides an implementation of the PortLocator's abstract connectPort() method that can connect a ModemPort that has been acquired by the PortLocator base class. This derived class logs information in the comm failure database table relating to connection problems that may occur.

5.19.1.1.6 PolledTSSImpl (Class)

This object implements the Transportation Sensor System interface as defined in IDL. This implementation provides the base functionality required for Transportation Sensor Systems that are polled periodically to retrieve traffic parameters. The only requirement for derived classes is to provide an implementation of the abstract poll method, which communicates over a previously connected Port to obtain the traffic parameters from a TSS.

This implementation periodically polls the field device using the derived class implementation of the poll method. This implementation provides services such as raw data logging, averaging/summation of data into configured zone groups, asynchronous notification of configuration changes, and persistence/depersistence.

A DeviceFailure alert is created each time the device transitions into HARDWARE_FAILURE. Devices that cycle in and out of HARDWARE_FAILURE will send multiple DeviceFailure alerts so it is up to the AlertModule to prevent duplicate open DeviceFailure alerts for the same device.

5.19.1.1.7 PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.19.1.1.8 RTMS (Class)

The Remote Traffic Microwave Sensor (RTMS) is a detector manufactured by EIS, Inc. capable of providing lane level volume, speed, and occupancy data for up to 8 lanes of a roadway at a single location. This interface serves to identify TransportationSensorSystem objects as being of the type RTMS. It also provides a place holder for future operations that may not apply to TSS objects in general and are instead RTMS specific.

5.19.1.1.9 RTMSDeviceStatus (Class)

This class is used to pass raw data retrieved from the RTMS to the caller of the RTMSProtocolHdlr getStatus() method.

m_trafficParameters - the traffic parameters sensed by the device, such as volume, speed, and occupancy.

m_healthStatus - The health status byte reported from the RTMS. A value other than 10, 20, 30, 40, 50, 60, or 70 indicates a hardware problem.

m_msgNum - The message number reported by the RTMS. This number is incremented sequentially when the RTMS dumps averaged data to a retrieval area at the end of a message period. It can be used to determine if the device is being polled too frequently or infrequently.

5.19.1.1.10 RTMSFactoryImpl (Class)

This class implements the RTMSFactory interface as defined in the IDL. It holds all RTMSImpl objects that have been created within an instance of the RTMSManagementModule and allows for the addition and removal of RTMS objects. It also allows one to query all RTMS objects currently served from the factory.

This factory contains a timer that periodically fires, causing the RTMSFactoryImpl to collect the current status of each RTMSImpl and push the collective status in a single CORBA event.

5.19.1.1.11RTMSImpl (Class)

This class is a derivation of the PolledTSSImpl that provides functionality for obtaining the current traffic parameters from an RTMS device. It makes use of an RTMSProtocolHandler to perform the device specific protocol to obtain the traffic parameters. It moves the data from the device specific format to the generic TSSPollResults object to allow the PolledTSSImpl to combine/average data based on zone group configuration, perform raw data logging, and other services that are common to Transportation Sensor System objects.

5.19.1.1.12RTMSProtocolHdlr (Class)

This class is a utility that encapsulates the communication protocol of the RTMS device. It provides a high level method to get the status as an object. It formats a command and sends it to the device and receives and interprets the response from the device, passing the data back to the caller in the form of an RTMSDeviceStatus object.

5.19.1.1.13TransportationSensorSystem (Class)

A Transportation Sensor System (TSS) is a generic term used to describe a class of technology used for detection within the transportation industry. Examples of TSS devices range from the advanced devices, such as RTMS, to basic devices, such as single loop detectors.

This software interface is implemented by objects that provide access to the traffic parameters sensed by a Transportation Sensor System. Transportation Sensor Systems are capable of providing detection for one or more detection zones. A single loop detector would have one detection zone, while an RTMS would have 8 detection zones.

5.19.1.1.14TSSConfiguration (Class)

This class holds configuration data for a transportation sensor system (TSS) as follows:

m_id - The unique identifier for this TSS. This field is ignored when the object is passed to the TSS to change its configuration.

m_name - The name used to identify the TSS.

m_location - A descriptive location of the TSS.

m_dropAddress - The drop address for the device.

m_zoneGroups - Logical groupings of detection zones, used to provide a single set of traffic parameters for one or more detection zones.

m_pollIntervalSecs - The interval on which the TSS should be polled for its current traffic parameters (in seconds).

m_commPortCfg - Communication configuration values.

m_portLocData - Configuration information that determines which port manager(s) should be used to establish a connection with the SensorSystem.

m_debugComms - Flag used to enable/disable the logging of communications data for this TSS. When enabled, command and response packets exchanged with the device are logged to a debugging log file.

5.19.1.1.15TSSDBData (Class)

This class holds data that is retrieved from the database during start-up for a Transportation Sensor System object that existed in the system during a prior run of the software.

5.19.1.1.16TSSEvent (Class)

This class is a CORBA union that contains varying data depending on the current value of the discriminator.

If the discriminator is ConfigChanged, this union contains a TSSConfig object.

If the discriminator is ObjectAdded, this union contains an ObjectAddedEventInfo object.

If the discriminator is ObjectRemoved, this union contains a byte[] containing the unique identifier for the Traffic Sensor System that was removed.

If the discriminator is CurrentStatus the union contains an array of one or more TSSStatus objects.

If the discriminator is ModeChanged, the union contains a ModeChangedEventInfo.

If the discriminator is OpStatusChanged, the union contains an OpStatusChangedEventInfo object.

5.19.1.1.17TSSManagementDB (Class)

This class is a utility that provides methods for adding, removing, and updating database data pertaining to Transportation Sensor Systems. Because this class is designed to be generic and work for RTMS as well as other TSS derived objects, the add method requires a model id to be passed. This allows data for a specific model to be retrieved by model specific factories during system initialization.

5.19.1.1.18TSSPollingTask (Class)

This class is a TimerTask that is used by an RTMS to schedule its asynchronous polling with a Timer object.

5.19.1.1.19TSSPollResults (Class)

This class is a data holder used to pass the results of device polling from the PolledTSSImpl derived class back to the base class for processing. The traffic parameter data passed is lane (detection zone) level. The operational status is the status as determined by the derived class.

m_trafficParms - An array of traffic parameters for the current poll cycle, with one array entry for each detection zone of the device.

m_opStatus - The operational status as determined by the derived class.

5.19.1.1.20TSSStatus (Class)

This class holds current status information for a TSS as follows:

m_id - The ID of the TSS for which this status applies.

m_zoneGrpTrafficParms - The traffic parameters for each ZoneGroup of the Transportation Sensor System as specified in the Sensor system's TSSConfiguration object.

m_mode - The communication mode of the TSS.

m_opStatus - The operational status for the TSS.

m_trafficParameterTimestamp - A timestamp that records when the traffic parameter data was collected from the device.

m_avgSpeed - average speed at the detector level.

m_speedRange - speed range at the detector level (avg speed).

5.19.1.2 TSSManagementModulePkg (Class Diagram)

This package manages all server activities related to Traffic Sensor Systems. Currently only Remote Traffic Microwave Sensor (RTMS) type devices are supported however it is designed to handle other TSS devices types. Devices are periodically polled (responding to a device-created event is not supported) and results are reported on CORBA event channels.

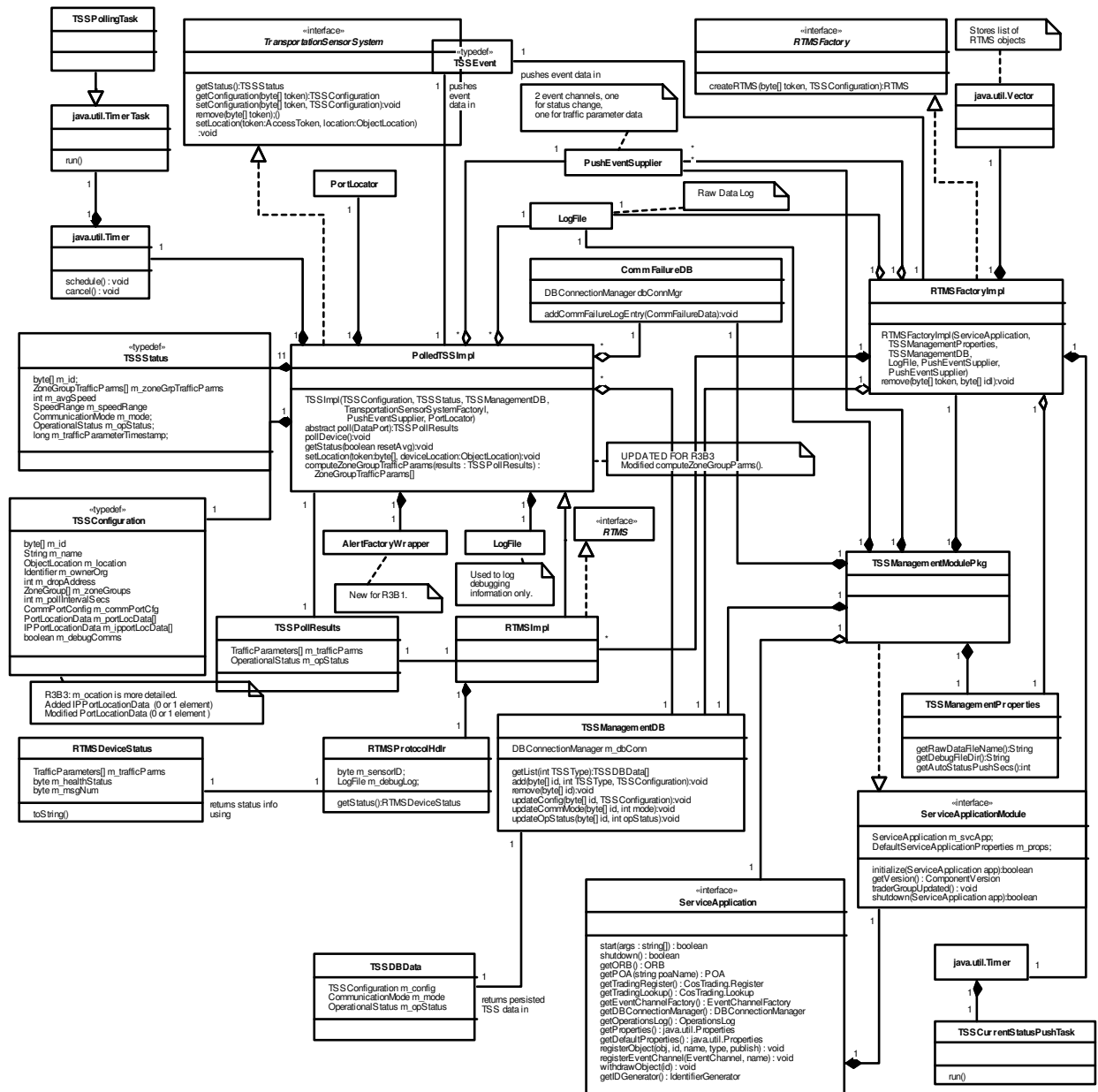


Figure 5-240. TSSManagementModulePkg (Class Diagram)

5.19.1.2.1 AlertFactoryWrapper (Class)

This singleton class provides a wrapper for the Alert Factory that provides automatic location of an Alert Factory and automatic re-discovery should the Alert Factory reference return an error. This class also allows for built-in fault tolerance by automatically failing over to a "working" Alert Factory without the user of this class being aware that this being done. In addition, this class defers the discovery of the Alert Factory until its first use, thus eliminating a start-up dependency for modules that use the Alert Factory.

This class delegates all of its method calls to the system AlertFactory using its currently known good reference to an AlertFactory. If the current reference returns a CORBA failure in the delegated call, this class automatically switches to another reference. When there are no good references (as is true the first time the object is used), this class issues a trader query to (re)discover the published Alert Factory objects in the system. During a method call, the trader will be queried at most one time and under normal circumstances, not at all.

5.19.1.2.2 CommFailureDB (Class)

This class is a utility used to log an entry in the Comm Failure log table in the database. This table is used to log details about any comm failure that occurs in the system.

5.19.1.2.3 java.util.Timer (Class)

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

5.19.1.2.4 java.util.TimerTask (Class)

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

5.19.1.2.5 java.util.Vector (Class)

A Vector is a growable array of objects.

5.19.1.2.6 LogFile (Class)

This class creates a flat file for writing system trace log messages and purges them at user specified interval. The log files created by this class are used for system debugging and maintenance only and are not to be confused with the system operations log which is modeled by the OperationsLog class.

5.19.1.2.7 PolledTSSImpl (Class)

This object implements the Transportation Sensor System interface as defined in IDL. This implementation provides the base functionality required for Transporation Sensor Systems that are polled periodically to retrieve traffic parameters. The only requirement for derived classes is to provide an implmentation of the abstract poll method, which communicates over a previously connected Port to obtain the traffic parameters from a TSS.

This implementation periodically polls the field device using the derived class implementation of the poll method. This implementation provides services such as raw data logging, averaging/summation of data into configured zone groups, asynchronous notification of configuration changes, and persistence/depersistence.

A DeviceFailure alert is created each time the device transitions into `HARDWARE_FAILURE`. Devices that cycle in and out of `HARDWARE_FAILURE` will send multiple DeviceFailure alerts so it is up to the AlertModule to prevent duplicate open DeviceFailure alerts for the same device.

5.19.1.2.8 PortLocator (Class)

The PortLocator is a utility class that helps one to connect to the port used by the device. The actual implementation of the operations is done by the derived classes depending on what protocol is used for communication.

5.19.1.2.9 PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.19.1.2.10 RTMS (Class)

The Remote Traffic Microwave Sensor (RTMS) is a detector manufactured by EIS, Inc. capable of providing lane level volume, speed, and occupancy data for up to 8 lanes of a roadway at a single location. This interface serves to identify `TransportationSensorSystem` objects as being of the type `RTMS`. It also provides a place holder for future operations that may not apply to TSS objects in general and are instead `RTMS` specific.

5.19.1.2.11 RTMSDeviceStatus (Class)

This class is used to pass raw data retrieved from the RTMS to the caller of the `RTMSProtocolHdlr getStatus()` method.

`m_trafficParameters` - the traffic parameters sensed by the device, such as volume, speed, and occupancy.

`m_healthStatus` - The health status byte reported from the RTMS. A value other than 10, 20, 30, 40, 50, 60, or 70 indicates a hardware problem.

m_msgNum - The message number reported by the RTMS. This number is incremented sequentially when the RTMS dumps averaged data to a retrieval area at the end of a message period. It can be used to determine if the device is being polled too frequently or infrequently.

5.19.1.2.12RTMSFactory (Class)

Objects which implement RTMSFactory are capable of adding an RTMS to the system.

5.19.1.2.13RTMSFactoryImpl (Class)

This class implements the RTMSFactory interface as defined in the IDL. It holds all RTMSImpl objects that have been created within an instance of the RTMSManagementModule and allows for the addition and removal of RTMS objects. It also allows one to query all RTMS objects currently served from the factory.

This factory contains a timer that periodically fires, causing the RTMSFactoryImpl to collect the current status of each RTMSImpl and push the collective status in a single CORBA event.

5.19.1.2.14RTMSImpl (Class)

This class is a derivation of the PolledTSSImpl that provides functionality for obtaining the current traffic parameters from an RTMS device. It makes use of an RTMSProtocolHandler to perform the device specific protocol to obtain the traffic parameters. It moves the data from the device specific format to the generic TSSPollResults object to allow the PolledTSSImpl to combine/average data based on zone group configuration, perform raw data logging, and other services that are common to Transportation Sensor System objects.

5.19.1.2.15RTMSProtocolHdlr (Class)

This class is a utility that encapsulates the communication protocol of the RTMS device. It provides a high level method to get the status as an object. It formats a command and sends it to the device and receives and interprets the response from the device, passing the data back to the caller in the form of an RTMSDeviceStatus object.

5.19.1.2.16ServiceApplication (Class)

This interface is implemented by objects that can provide the basic services needed by a ChartII service application. These services include providing access to basic CORBA objects that are needed by service applications, such as the ORB, POA, Trader, and Event Service.

5.19.1.2.17ServiceApplicationModule (Class)

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the

invoking ServiceApplication to perform actions such as object creation and publication.

5.19.1.2.18TransportationSensorSystem (Class)

A Transportation Sensor System (TSS) is a generic term used to describe a class of technology used for detection within the transportation industry. Examples of TSS devices range from the advanced devices, such as RTMS, to basic devices, such as single loop detectors.

This software interface is implemented by objects that provide access to the traffic parameters sensed by a Transportation Sensor System. Transportation Sensor Systems are capable of providing detection for one or more detection zones. A single loop detector would have one detection zone, while an RTMS would have 8 detection zones.

5.19.1.2.19TSSConfiguration (Class)

This class holds configuration data for a transportation sensor system (TSS) as follows:

m_id - The unique identifier for this TSS. This field is ignored when the object is passed to the TSS to change its configuration.

m_name - The name used to identify the TSS.

m_location - A descriptive location of the TSS.

m_dropAddress - The drop address for the device.

m_zoneGroups - Logical groupings of detection zones, used to provide a single set of traffic parameters for one or more detection zones.

m_pollIntervalSecs - The interval on which the TSS should be polled for its current traffic parameters (in seconds).

m_commPortCfg - Communication configuration values.

m_portLocData - Configuration information that determines which port manager(s) should be used to establish a connection with the SensorSystem.

m_debugComms - Flag used to enable/disable the logging of communications data for this TSS. When enabled, command and response packets exchanged with the device are logged to a debugging log file.

5.19.1.2.20TSSCurrentStatusPushTask (Class)

This class is a timer task that is executed on a regular interval. When this task is run, it calls into the RTMSFactoryImpl object to have it collect the status for all RTMSImpl

objects and to push a CurrentStatus event with the collected data.

5.19.1.2.21 TSSDBData (Class)

This class holds data that is retrieved from the database during start-up for a Transportation Sensor System object that existed in the system during a prior run of the software.

5.19.1.2.22 TSSEvent (Class)

This class is a CORBA union that contains varying data depending on the current value of the discriminator.

If the discriminator is ConfigChanged, this union contains a TSSConfig object.

If the discriminator is ObjectAdded, this union contains an ObjectAddedEventInfo object.

If the discriminator is ObjectRemoved, this union contains a byte[] containing the unique identifier for the Traffic Sensor System that was removed.

If the discriminator is CurrentStatus the union contains an array of one or more TSSStatus objects.

If the discriminator is ModeChanged, the union contains a ModeChangedEventInfo.

If the discriminator is OpStatusChanged, the union contains an OpStatusChangedEventInfo object.

5.19.1.2.23 TSSManagementDB (Class)

This class is a utility that provides methods for adding, removing, and updating database data pertaining to Transportation Sensor Systems. Because this class is designed to be generic and work for RTMS as well as other TSS derived objects, the add method requires a model id to be passed. This allows data for a specific model to be retrieved by model specific factories during system initialization.

5.19.1.2.24 TSSManagementModulePkg (Class)

This class is a ServiceApplicationModule used to serve an RTMSFactory object. The RTMSFactory serves zero or more RTMS objects. By providing an implementation of the ServiceApplicationModule interface, this class can be included in the CHART2 service application framework, which provides common services needed to serve CORBA objects within the CHART 2 system.

5.19.1.2.25 TSSManagementProperties (Class)

This class provides a wrapper to the application's properties file that provides easy access to the properties specific to the TSSManagementModule. These properties include the name of the file where raw traffic parameter data is to be logged, the directory where debug log files are to be kept, and the interval at which the status of all TSS objects is to be collected

and pushed in a CORBA event.

5.19.1.2.26TSSPollingTask (Class)

This class is a TimerTask that is used by an RTMS to schedule its asynchronous polling with a Timer object.

5.19.1.2.27TSPollResults (Class)

This class is a data holder used to pass the results of device polling from the PolledTSSImpl derived class back to the base class for processing. The traffic parameter data passed is lane (detection zone) level. The operational status is the status as determined by the derived class.

m_trafficParms - An array of traffic parameters for the current poll cycle, with one array entry for each detection zone of the device.

m_opStatus - The operational status as determined by the derived class.

5.19.1.2.28TSSStatus (Class)

This class holds current status information for a TSS as follows:

m_id - The ID of the TSS for which this status applies.

m_zoneGrpTrafficParms - The traffic parameters for each ZoneGroup of the Transportation Sensor System as specified in the Sensor system's TSSConfiguration object.

m_mode - The communication mode of the TSS.

m_opStatus - The operational status for the TSS.

m_trafficParameterTimestamp - A timestamp that records when the traffic parameter data was collected from the device.

m_avgSpeed - average speed at the detector level.

m_speedRange - speed range at the detector level (avg speed).

5.19.2 SequenceDiagrams

5.19.2.1 PolledTSSImpl:computeZoneGroupTrafficParms (Sequence Diagram)

This diagram depicts changes to the `PolledTSSImpl.computeZoneGroupTrafficParms()` method needed to support zone level detail data and TSS summary data (speed range) enhancements for R3B3. Existing implementation details are diagrammed at a high level or summarized using imbedded notes. This method uses a `TSSPollResults` object which contains zone level data to create an array of `ZoneGroupTrafficParms` objects to return to the caller. First, system wide `SpeedRange` values are retrieved from the `SystemProfileProperties`. For each zone group, an array of `ZoneTrafficParms` objects, representing zone level detail in R3B3, is created including new `SpeedRange` values. Each `ZoneGroup` also has a `ZoneGroup` level `SpeedRange` for R3B3.

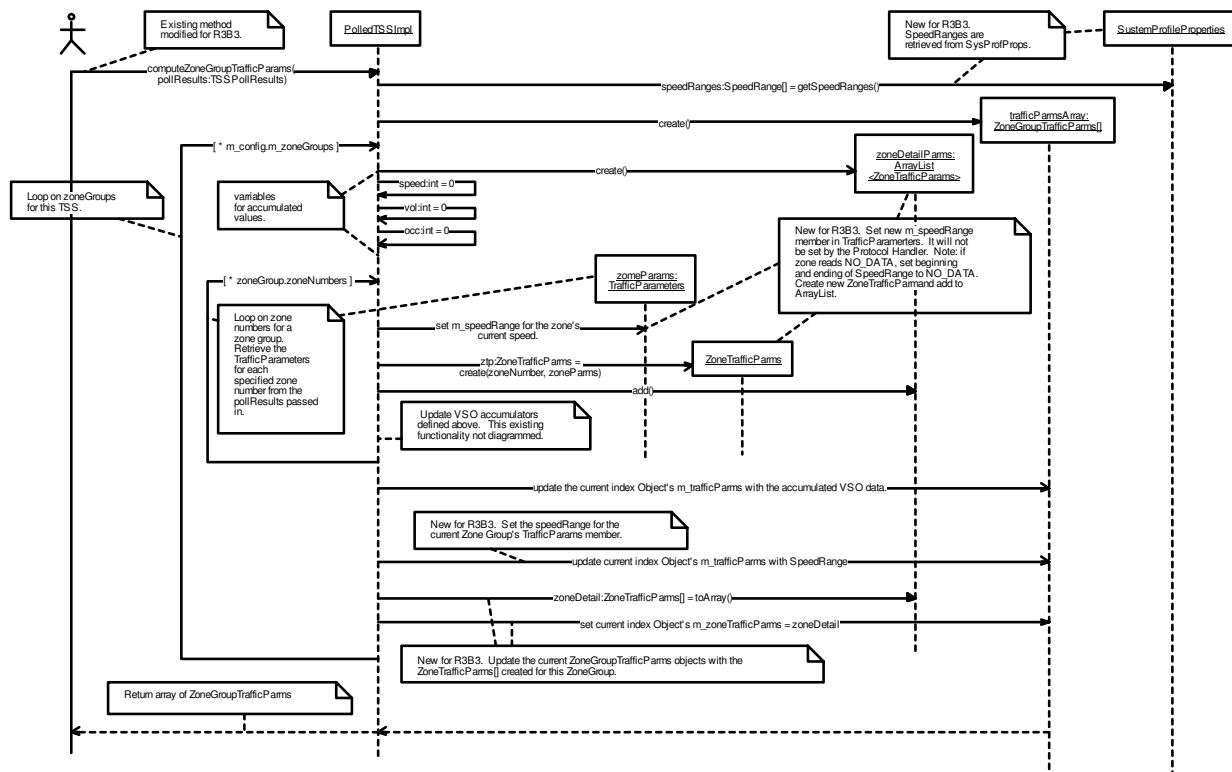


Figure 5-241. PolledTSSImpl:computeZoneGroupTrafficParms (Sequence Diagram)

5.19.2.2 PolledTSSImpl:processPollResults (Sequence Diagram)

This diagram depicts changes to the PolledTSSImpl.processPollResults() method needed to support zone level detail data and TSS summary data (speed range) enhancements for R3B3. Existing implementation details are diagrammed at a high level or summarized using imbedded notes. This method processes the TSSPollResults returned from the protocol handler as the result of a poll. If the TSSPollResults indicates the op status is OK, the following processing is done. First, raw zone level detail is logged to a separate file. Then the computeZoneGroupTrafficParms() method is called and returns an array of ZoneGroupTrafficParms. The returned array is used to update the current TSSStatus. Then the zone group data is used to calculate and average speed (TSSStatus.m_avgSpeed). Previously the average speed was calculated on the GUI.

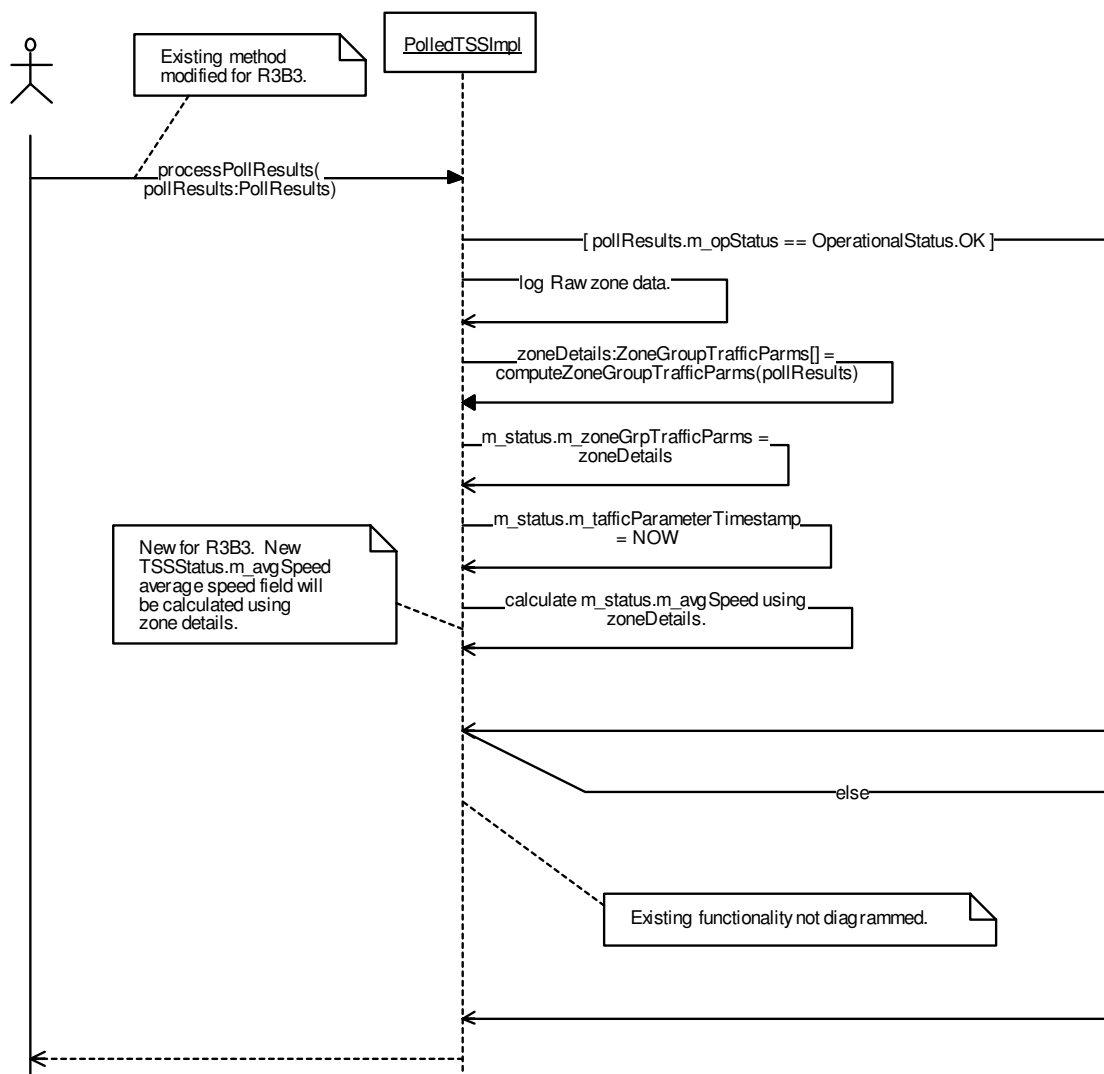


Figure 5-242. PolledTSSImpl:processPollResults (Sequence Diagram)

5.19.2.3 PolledTSSImpl:setConfiguration (Sequence Diagram)

A user with the proper functional rights can change the configuration of a Transportation Sensor System. The previous configuration values are used to detect values that have been changed. If the Port location data has been changed, a new PortLocator object is created with the new values. If the polling interval has been changed and the device is not offline, the existing polling timer is cancelled and destroyed, a new timer is created, and a new polling task is scheduled. If any values were changed, an entry is made in the operations log to record the values that the user has changed. A CORBA event is pushed on the Status event channel to provide notification of the configuration change to other applications.

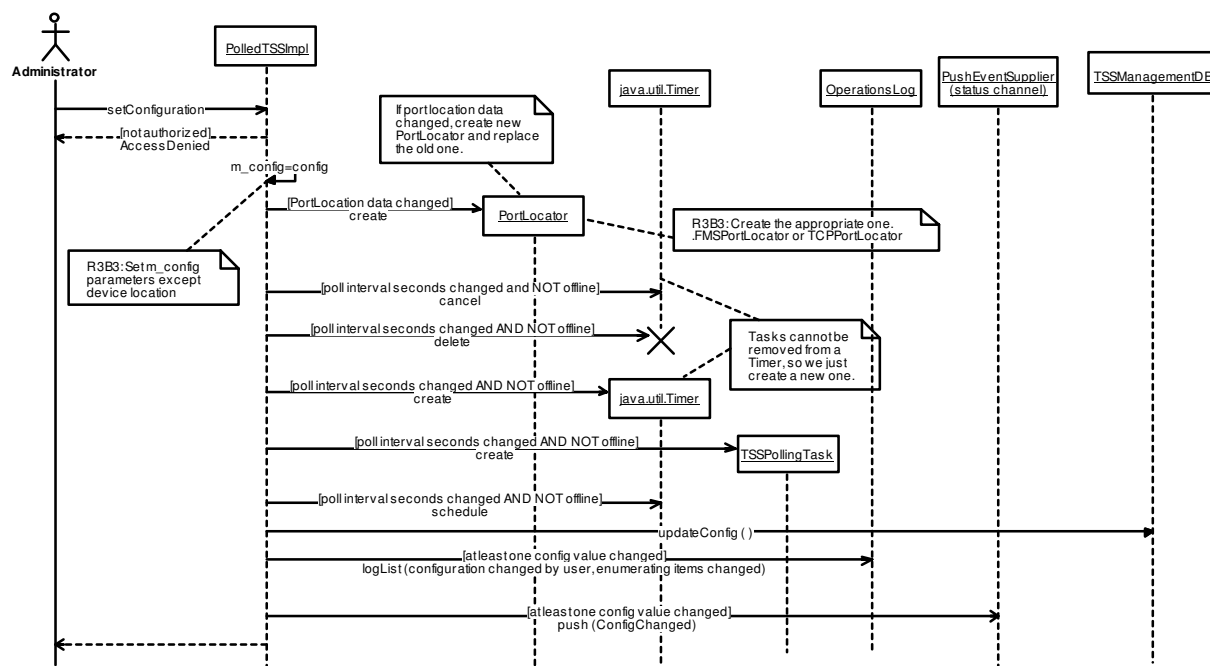


Figure 5-243. PolledTSSImpl:setConfiguration (Sequence Diagram)

5.19.2.4 RTMSFactoryImpl:constructor (Sequence Diagram)

When the RTMSFactoryImpl is constructed, it obtains persisted data for each previously existing RTMS from the database and constructs RTMSImpl objects using this data. Each object is connected to the ORB and registered in the CORBA trading service. The factory creates a timer that is used to cause it to periodically collect the status of all RTMS objects and push the data as a CORBA event.

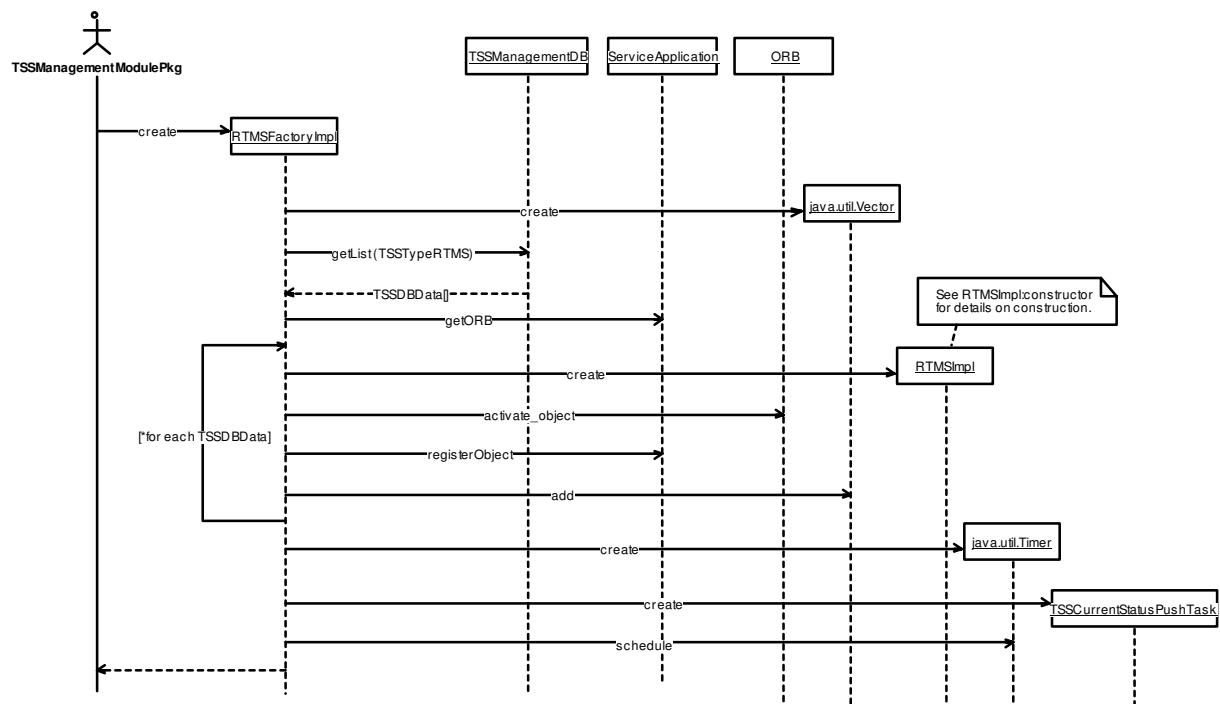


Figure 5-244. RTMSFactoryImpl:constructor (Sequence Diagram)

5.19.2.5 RTMSImpl:constructor (Sequence Diagram)

This diagram shows the construction of the RTMSImpl object. The RTMSImpl invokes the base class constructor, allowing it to construct a PortLocator, LogFile (for debugging), and a polling timer (if the status passed to the constructor does not indicate the device is offline). After the base class is constructed, the RTMSImpl constructs an RTMSProtocolHandler to be used to perform the RTMS specific protocol to obtain traffic parameters from the RTMS device.

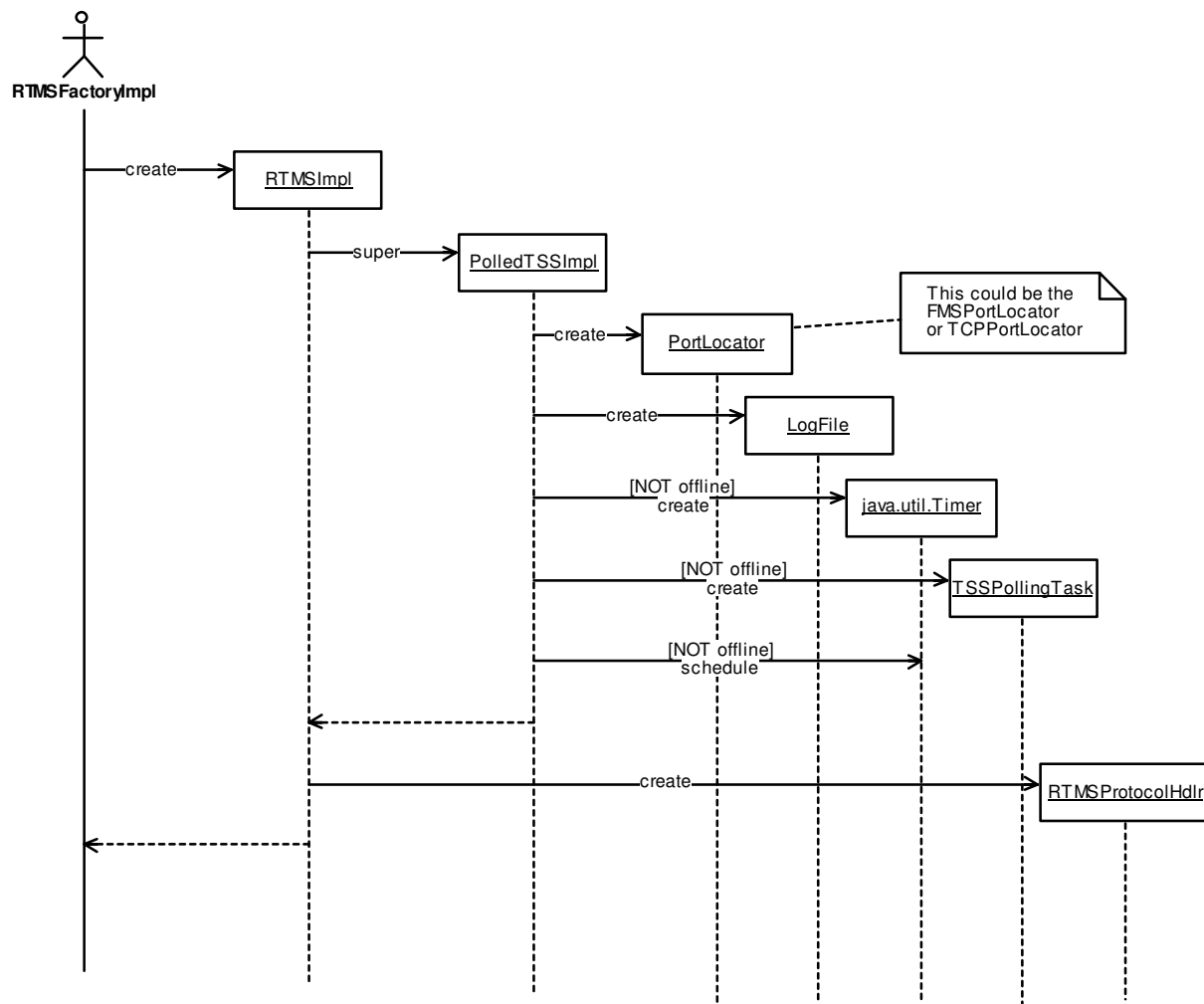


Figure 5-245. RTMSImpl:constructor (Sequence Diagram)

5.19.2.6 RTMSImpl:poll (Sequence Diagram)

The poll method of the RTMSImpl is called from its base class when it is time to poll the RTMS device. At the point when this method is called, the base class has already established a connection with the device. The RTMSImpl uses the RTMSProtocolHandler to send a data request to the device and parse the device response. Any communication failure, such as a non-responsive device, causes the base class to be notified that a communication failure occurred. If a communication failure did not occur, the RTMS health status is checked for an indication of a hardware failure. If no hardware failure exists, the lane level data is passed back to the base class to process the data.

A DeviceFailureAlert is created only when the RTMS transitions from another state into **HARDWARE_FAILURE**. Any future transitions into another state have no effect on the alert. A device that cycles in and out of a hardware failure causes this class to generate many DeviceFailure alerts however it is left to the AlertModule to not create duplicate open alerts.

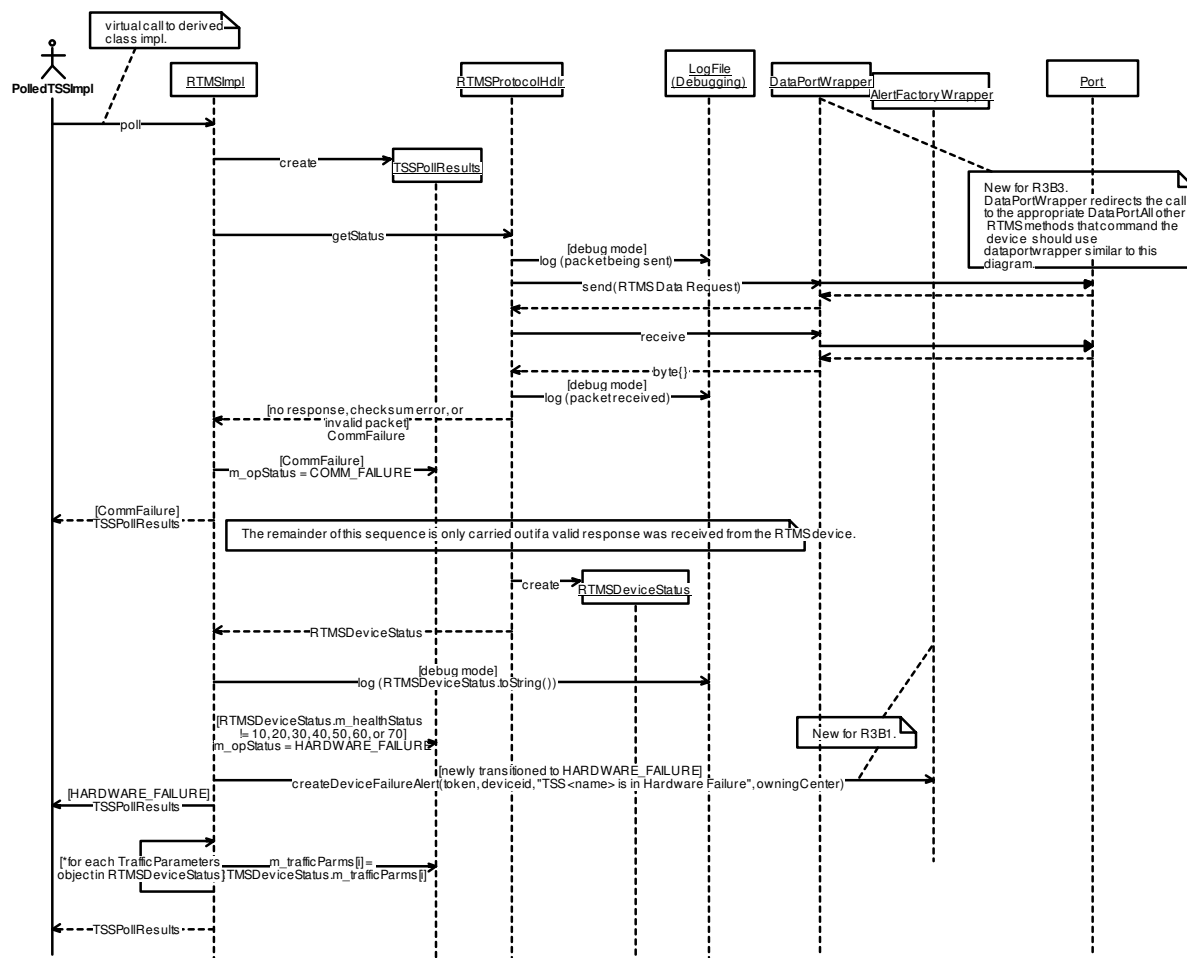


Figure 5-246. RTMSImpl:poll (Sequence Diagram)

5.20 TravelRouteModulePkg

5.20.1 Classes

5.20.1.1 TravelRouteModule (Class Diagram)

This Class Diagram shows the classes involved in implementation of the Travel Route Module. Key classes include the TravelRouteModule, the TravelRouteFactory, and the TravelRouteImpl. There will be one module, one factory, and many TravelRouteImpl objects.



This singleton class provides a wrapper for the Alert Factory that provides automatic location of an Alert Factory and automatic re-discovery should the Alert Factory reference

return an error. This class also allows for built-in fault tolerance by automatically failing over to a "working" Alert Factory without the user of this class being aware that this is being done. In addition, this class defers the discovery of the Alert Factory until its first use, thus eliminating a start-up dependency for modules that use the Alert Factory.

This class delegates all of its method calls to the system AlertFactory using its currently known good reference to an AlertFactory. If the current reference returns a CORBA failure in the delegated call, this class automatically switches to another reference. When there are no good references (as is true the first time the object is used), this class issues a trader query to (re)discover the published Alert Factory objects in the system. During a method call, the trader will be queried at most one time and under normal circumstances, not at all.

5.20.1.1.2 Chart2DMS (Class)

The Chart2DMS class extends the DMS interface and defines a more detailed interface to be used in manipulating the CHART-specific DMS objects within CHART. It provides an interface for traffic events to provide input as to what each traffic event desires to be on the sign via the ArbitrationQueue interface. Through the HARMessageNotifier interface, a HAR can use a DMS to notify travelers to tune in to a radio station to hear a traffic message. CHART business rules include concepts such as shared resources, arbitration queues, and linking device usage to traffic events. These concepts go beyond industry-standard DMS control. This includes an ability to enable and disable CHART traveler information messages, which were added in R3B3.

5.20.1.1.3 CommandQueue (Class)

The CommandQueue class provides a queue for QueueableCommand objects. The CommandQueue has a thread that it uses to process each QueueableCommand in a first in first out order. As each command object is pulled off the queue by the CommandQueue's thread, the command object's execute method is called, at which time the command performs its intended task.

5.20.1.1.4 ConsumerPushCmd (Class)

This class is used to queue pushing an update to a TravelRouteConsumer for asynchronous execution on a Command Queue. The pusher CommandQueue runs with many threads (e.g., the default of 10), so that delays in connecting to an unreachable consumer do not much delay getting the pushes out to the consumers that are reachable.

5.20.1.1.5 DBConnectionManager (Class)

This class implements a database connection manager that manages a pool of database connections. Any CHART II system thread requiring database access gets a database connection from the pool of connections maintained by this manager class. The connections are maintained in two separate lists namely, inUseList and freeList. The inUseList contains connections that have already been assigned to a thread. The freeList contains unassigned connections. This class assumes that an appropriate JDBC driver has been loaded either by using the "jdbc.drivers" system property or by loading it explicitly. The class has a monitor

thread that is started by the constructor. This connection monitor thread periodically checks the inuseList to see if there are connections that are owned by dead threads and move such connections to the freeList. The connection monitor thread is started only if a non-zero value is specified for the monitoring time interval in the constructor.

5.20.1.1.6 ExtLinkIdKey (Class)

This structure is an ExtLinkSpec holder object used by the Travel Route Module as a hashable ExtLinkSpec object which can be used as the key into a Hashtable of external roadway link specs. It is used as the Hashtable key in a table of TravelRouteImpl objects organized by ExtTollSpec.

5.20.1.1.7 ExtLinkSpec (Class)

This structure is used to hold all the identifying information about a link as known to an external system (link data supplier).

5.20.1.1.8 ExtTollSpec (Class)

This structure is used to identify a toll rate route. It contains the supplying external system, the start ID and end ID of the toll rate route (which is the "key" used to identify the toll rate route, and the name by which the external system refers to the route.

5.20.1.1.9 ExtTollSpecKey (Class)

This structure is an ExtTollSpec holder object used by the Travel Route Module as a hashable ExtTollSpec object which can be used as the key into a Hashtable of external toll rate specs. It is used as the Hashtable key in a table of UsedLinkElement objects organized by ExtLinkId.

5.20.1.1.10 java.util.Properties (Class)

The Properties class represents a persistent set of properties. The Properties can be saved to a stream or loaded from a stream. Each key and its corresponding value in the property list is a string. A property list can contain another property list as its "defaults"; this second property list is searched if the property key is not found in the original property list.

5.20.1.1.11 java.util.Timer (Class)

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

5.20.1.1.12 java.util.TimerTask (Class)

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

5.20.1.1.13LinkTravTimeHistRecord (Class)

This structure is used to store the most recent X historical link travel time data points acquired by the system. This is a circular array, with the head (oldest record) referenced in the LinkTravelTimeHistStats. The tail (newest record) is head-1 (mod X), and is always the same data point as is contained in the LinkTravelTimeStats structure.

5.20.1.1.14LinkTravTimeHistStats (Class)

This structure (together with the LinkTravelTimeHistRecord) stores the most recent X historical link travel time data points. It contains the head pointer and the LinkTravelTimeHistRecord array which contains the actual data. This structure is not pushed on the event channel, because it normally contains data which clients could have already accumulated themselves, but is available on demand.

5.20.1.1.15LinkTravTimeStats (Class)

This structure contains the most recent travel time data point acquired for a roadway link. It matches the most recent record in the LinkTravelTimeHistStats.

5.20.1.1.16ProcessLinkDataCmd (Class)

This is a QueueableCommand placed on the processor CommandQueue to process incoming roadway link statistical data sent in from a link data supplier. The QueueableCommand is used to move the processing off of the incoming CORBA processing loop so that return can be returned quickly to the LinkDataProvider caller.

5.20.1.1.17ProcessTollDataCmd (Class)

This is a QueueableCommand placed on the processor CommandQueue to process incoming toll rate statistical data sent in from a toll rate data supplier. The QueueableCommand is used to move the processing off of the incoming CORBA processing loop so that return can be returned quickly to the TollDataProvider caller.

5.20.1.1.18PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.20.1.1.19QueueableCommand (Class)

A QueueableCommand is an interface used to represent a command that can be placed on a CommandQueue for asynchronous execution. Derived classes implement the execute method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a CommandQueue. The CommandQueue driver calls the execute method to execute a command in the queue and a call to the interrupted method is made when a CommandQueue is shut down.

5.20.1.1.20RoadwayLinkConfig (Class)

This structure contains the configuration data for a roadway link. It includes the external system name (e.g., "INRIX"), the ID by which the external system identifies the link, and location data.

5.20.1.1.21RoadwayLinkConfigInfo (Class)

This is a convenience structure which combines a roadway link ID with the RoadwayLinkConfig. It is used for passing configuration data about all links in the system or all links in one route from the Travel Route Module to clients.

5.20.1.1.22RoadwayLinkFullInfo (Class)

This is a convenience structure which combines a roadway link ID with its configuration, current, and historical stats. It is used for passing the full set of data for all links in the system or all links in one route from the Travel Route Module to clients.

5.20.1.1.23RouteAndLinkConfig (Class)

This convenience structure is used to pass configuration data for a travel route plus configuration data for all the route's links from the Travel Route Module to clients.

5.20.1.1.24RouteFullStats (Class)

This convenience structure holds all stats data for a single travel route -- current stats and recent historical stats for the route and for the constituent links (if any).

5.20.1.1.25RouteHistStats (Class)

This structure is used to store historical statistical data (travel times and toll rates) for a Travel Route. It consists of an array of zero or one RouteTravelTimeHistStats, storing Travel Time historical statistics (if configured with links), and an array of zero or one RouteTollRateHistStats, storing Toll Rate historical statistics (if configured to track toll rates).

5.20.1.1.26RouteLink (Class)

This structure makes the association between a travel route and one roadway link which helps comprise the route, together with parameters associated with the use of the link within that particular route: the percent of the link to include in the route, and the minimum acceptable quality for link travel time data as used in that particular route.

5.20.1.1.27RouteLinkHistStats (Class)

This convenience structure contains historical travel time stats for a travel route and for all the links which comprise the route. It is available on demand from TravelRoute.

5.20.1.1.28RouteLinkStats (Class)

This convenience structure contains current travel time stats for a travel route and for all the links which comprise the route. It is available on demand from TravelRoute.

5.20.1.1.29RouteStats (Class)

This convenience structure combines the current travel time data and toll rate data for a travel route. (It contains zero or one of each, depending on what types of data the route is configured to track.) It is available on demand via the TravelRoute.

5.20.1.1.30RouteTollRateHistRecord (Class)

This structure is used to store the most recent X historical route toll rate data points accumulated by the system. This is a circular array, with the head (oldest record) referenced in the RouteTollRateHistStats. The tail (newest record) is head-1 (mod X), and is always the same data point as is contained in the RouteTollRateStats structure.

5.20.1.1.31RouteTollRateHistStats (Class)

This structure (together with the RouteTollRateHistRecord) stores the most recent X historical route toll rate data points. It contains the head pointer and the RouteTollRateHistRecord array which contains the actual data. This structure is not pushed on the event channel, because it normally contains data which clients could have already accumulated themselves, but is available on demand.

5.20.1.1.32RouteTollRateStats (Class)

This structure contains the current toll rate data for a travel route. This includes the time the rate became effective and the toll rate itself. This data is also provided in the most recent entry in the history structure. The toll rate field may contain a negative number defined by StatsConstants, which indicates an error. There are two other fields NOT provided in the history structure -- the time the toll rate expires, and a reason string. This will be the empty string if the toll rate has been successfully provided recently, or details on the error condition if an error constant is specified.

5.20.1.1.33RouteTravTimeHistRecord (Class)

This structure is used to store the most recent X historical route travel time data points accumulated by the system. This is a circular array, with the head (oldest record) referenced in the RouteTravelTimeHistStats. The tail (newest record) is head-1 (mod X), and is always the same data point as is contained in the RouteTravelTimeStats structure.

5.20.1.1.34RouteTravTimeHistStats (Class)

This structure (together with the RouteTravelTimeHistRecord) stores the most recent X historical route travel time data points. It contains the head pointer and the RouteTravelTimeHistRecord array which contains the actual data. This structure is not pushed on the event channel, because it normally contains data which clients could have already accumulated themselves, but is available on demand.

5.20.1.1.35RouteTravTimeStats (Class)

This structure contains the current travel time data for a travel route. This includes the time the travel time was computed, and the computed speed. This data is also provided in the most recent entry in the history structure. The travel time may contain a negative number defined by StatsConstants, which indicates an error. There are two other fields NOT provided in the history structure -- a computed trend (UP, DOWN, or FLAT) and a reason string. This will be the travel time calculation if it has been computed successfully, or details on the error condition if an error constant is specified.

5.20.1.1.36ServiceApplicationModule (Class)

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

5.20.1.1.37TollRateConfig (Class)

This structure holds the part of the Travel Route configuration pertaining to toll rates, if the travel route is configured to track toll rates. This structure holds the part of the Travel Route configuration pertaining to toll rates. One of these is contained in the TravelRouteConfig if the travel route is configured to track toll rates, otherwise there is none.

5.20.1.1.38TravelRoute (Class)

This is the primary CORBA interface for working with travel routes in CHART. This interface provides methods for getting various collections of configuration and/or statistical data for a travel route. It also provides methods for objects to register to be TravelRouteConsumer for the travel route (for instances, DMSs that have the route enabled in a traveler information message). Finally it provides methods for updating and removing travel routes.

5.20.1.1.39TravelRouteConfig (Class)

This structure holds the part of the Travel Route configuration pertaining to travel times, if the travel route is configured to track travel times. It contains the IDs of the links comprising the route, but not the link configurations themselves. (See RouteAndLinkConfig.)

5.20.1.1.40TravelRouteConsumer (Class)

This interface allows other CHART objects to register as a direct consumer of travel route statistical data. It provides operations for the travel route to call when the travel time or toll rate for the route is updated. A DMS registers as a TravelRouteConsumer when a TravelerInfoMsg is enabled.

5.20.1.1.41TravelRouteConsumerElement (Class)

This structure is used to hold a TravelRouteConsumer in a hashable object for use as a key in a Hashtable. It holds a reference to a TravelRouteConsumer (e.g., a DMS interested in receiving Travel Route updates for a given TravelRoute).

5.20.1.1.42TravelRouteConsumerInfo (Class)

This convenience structure lists a TravelRouteConsumer ID, reference, and type.

5.20.1.1.43TravelRouteDB (Class)

This class provides access to the database for the purpose of reading and writing TravelRoute information, including the travel routes themselves (configuration), and the statistical data (travel times and toll rates), as well as link configuration and statistical data.

5.20.1.1.44TravelRouteDisplayConfig (Class)

This structure stores that part of a TravelRoute configuration which a TravelRouteConsumer would be interested in, namely the travel route distance and the primary any alternate destination names. (The distance and destination may be required for display on a DMS travel route display, for instance.)

5.20.1.1.45TravelRouteFactory (Class)

This interface is the entry point for the Travel Route Management. It serves up travel routes (also an interface) and roadway links (structure data). It provides various operations for acquiring travel routes and roadway links. Since roadway links are not maintained as a separate interface, the factory provides the primary operations for acquiring link configuration and statistical data (although a TravelRoute interface also provides methods for acquiring such data about the links directly associated with it).

5.20.1.1.46TravelRouteFactoryImpl (Class)

This is the implementation class for the TravelRouteFactory interface.

5.20.1.1.47TravelRouteImpl (Class)

This is the implementation class for the TravelRoute interface.

5.20.1.1.48TravelRouteModule (Class)

This class is the CHART installable module used to maintain information on travel routes within CHART. It creates the TravelRouteFactoryImpl which stores the travel routes and roadway links.

5.20.1.1.49TravelRouteModuleProperties (Class)

This class provides access to properties stored in the properties files (.props file) for the TravelRouteModule.

5.20.1.1.50TravelRouteStalenessWatcherTask (Class)

This class implements the java.util.TimerTask interface and is placed on a java.util.Timer owned by the TravelRouteModule. It is used to ensure that travel time data does not become too stale, in the event that updates stop coming in from the link data provider. (Note that toll rates maintain come in supplying their own expiration time, so this task is concerned with travel times only.)

5.20.1.1.51TravTimeConfig (Class)

This structure holds the part of the Travel Route configuration pertaining to travel times. One of these is contained in the TravelRouteConfig if the travel route is configured to track travel times, otherwise there is none.

5.20.1.1.52UsedLinkElement (Class)

This class is used to match a roadway link to the travel routes which are currently configured to use it. These objects are stored in a Hashtable, keyed on ExtLinkIdKey (external link Ids), and which is used to efficiently find the Travel Routes which are configured to use a roadway link for which a travel time update is currently being processed.

5.20.2 SequenceDiagrams

5.20.2.1 TravelRouteDB:getLinks (Sequence Diagram)

This sequence diagram shows how links (configuration and current and recent historical statistics) are depersisted from the database during startup of the TravelRouteModule.

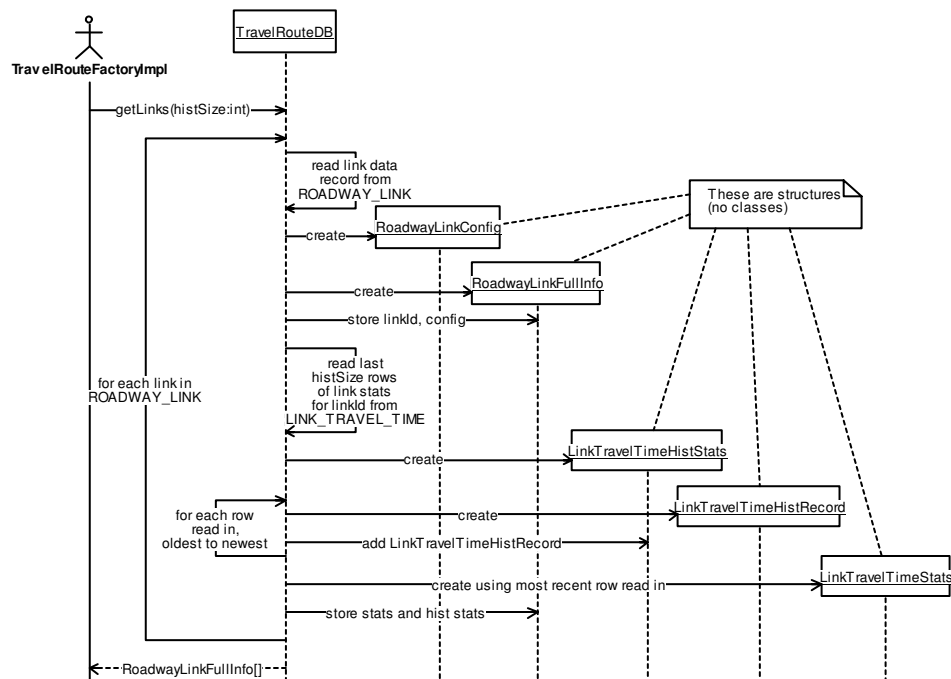


Figure 5-248. TravelRouteDB:getLinks (Sequence Diagram)

5.20.2.2 TravelRouteDB:getRoutes (Sequence Diagram)

This sequence diagram shows how travel routes (configuration and current and recent historical statistics) are depersisted from the database during startup of the TravelRouteModule.

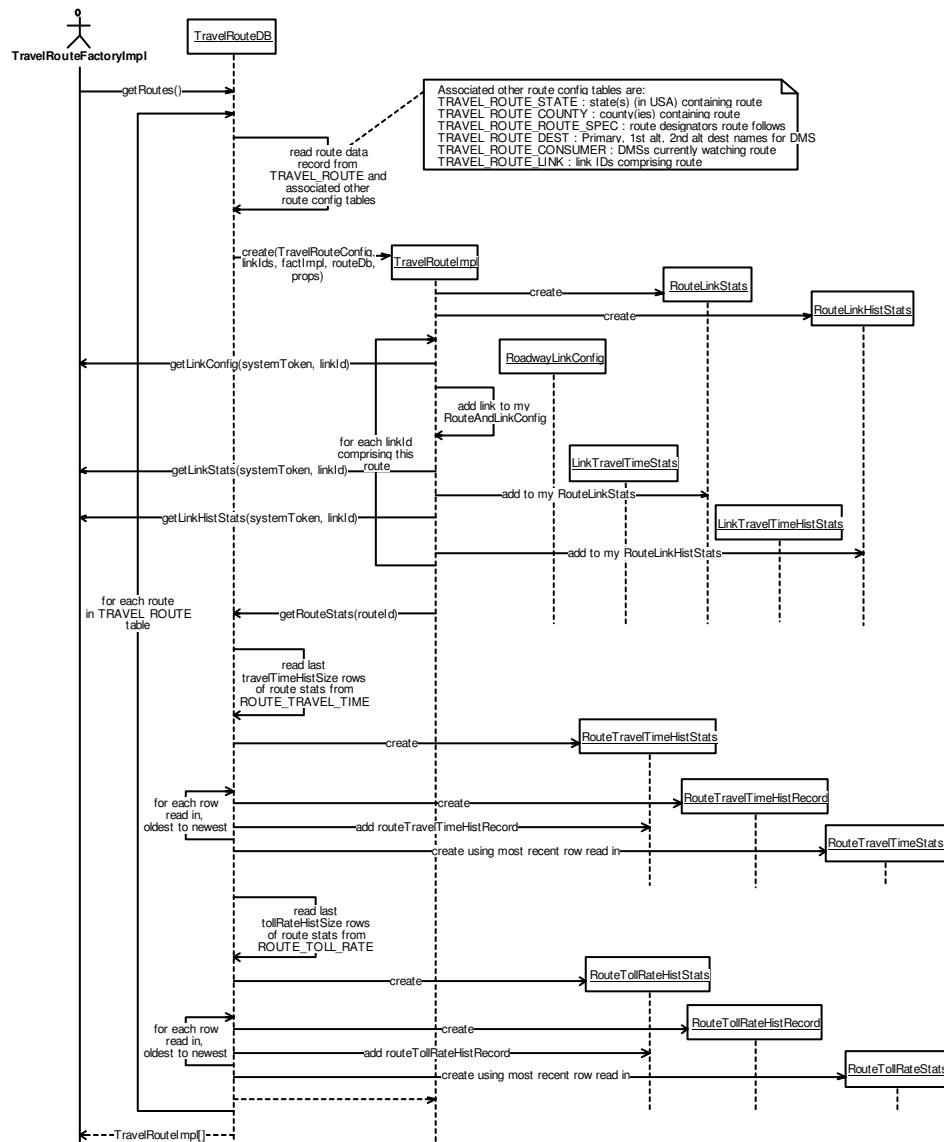


Figure 5-249. TravelRouteDB:getRoutes (Sequence Diagram)

5.20.2.3 TravelRouteFactoryImpl:addRoute (Sequence Diagram)

This sequence diagram shows the server side processing for adding a Travel Route to the

system. The GUI calls in with the new Route data. The server assigns an ID and inserts the route into the database and pushes the update on the Travel Route CORBA event channel, also returning the new route info directly to the caller.

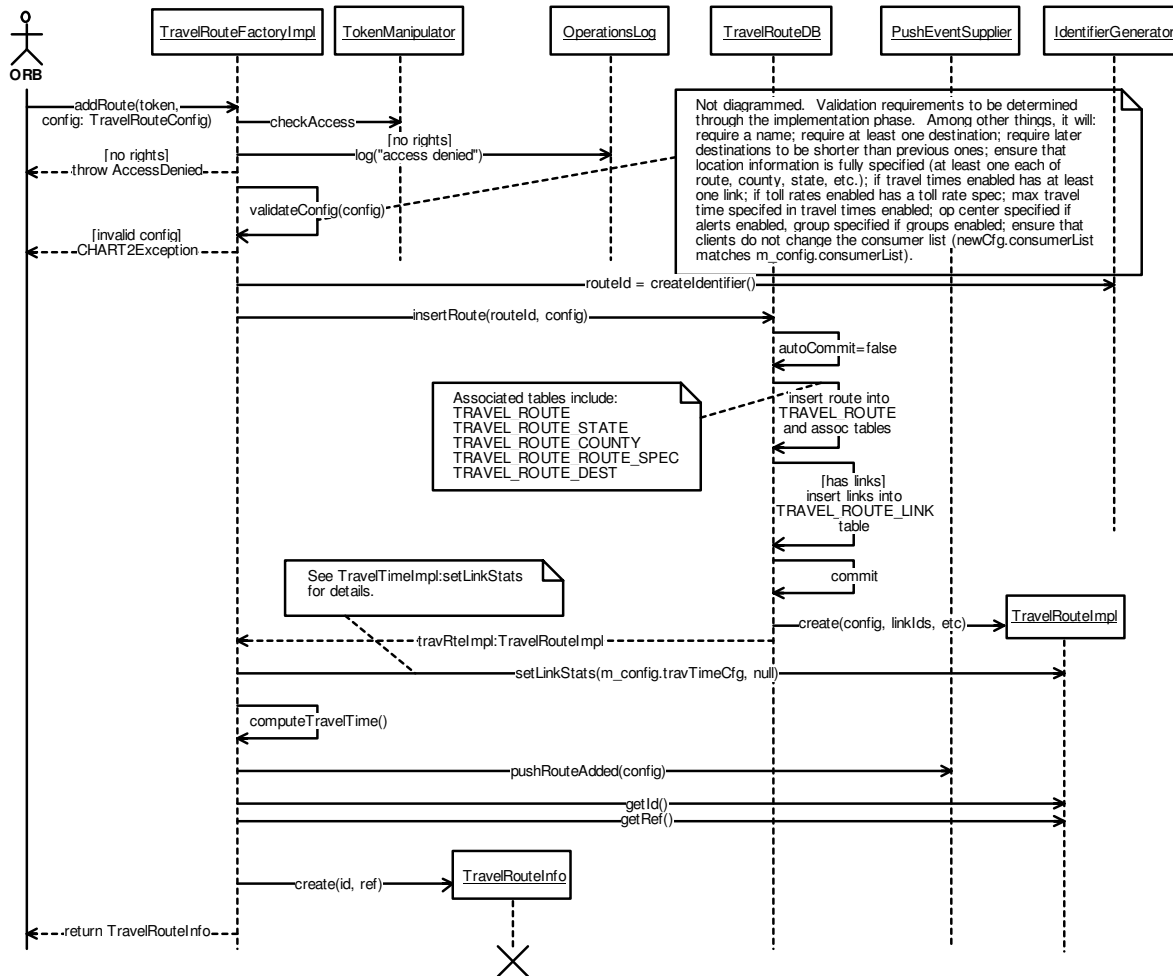


Figure 5-250. TravelRouteFactoryImpl:addRoute (Sequence Diagram)

5.20.2.4 TravelRouteFactoryImpl:computeTravelTime (Sequence Diagram)

This helper method is called by a TravelRouteImpl on itself to compute its own travel time. It is called whenever it is known that the underlying link data has changed, or when the link configuration of the route has changed such that the route travel time calculation would be different. Basically the specified percentage of each link's travel times are added together. Errors are tracked and the state code and string are updated after the main loop. The trend is computed, and the stats record is added to the history before returning.



5.20.2.5 TravelRouteFactoryImpl:pushConsumerUpdate (Sequence Diagram)

This diagram shows the four push methods that are called by a TravelRouteImpl (and one called by the TravelRouteFactoryImpl itself) on the TravelRouteFactoryImpl to push an update to a TravelRouteConsumer (e.g., a DMS which is using the route to feed an active Traveler Information Message). The initial call simply queues a ConsumerPushCmd on the factory's CommandQueue, and then the push call is made asynchronously. The pusher CommandQueue runs with many threads (e.g., the default of 10), so that delays in connecting to an unreachable consumer do not much delay getting the pushes out to the consumers that are reachable.

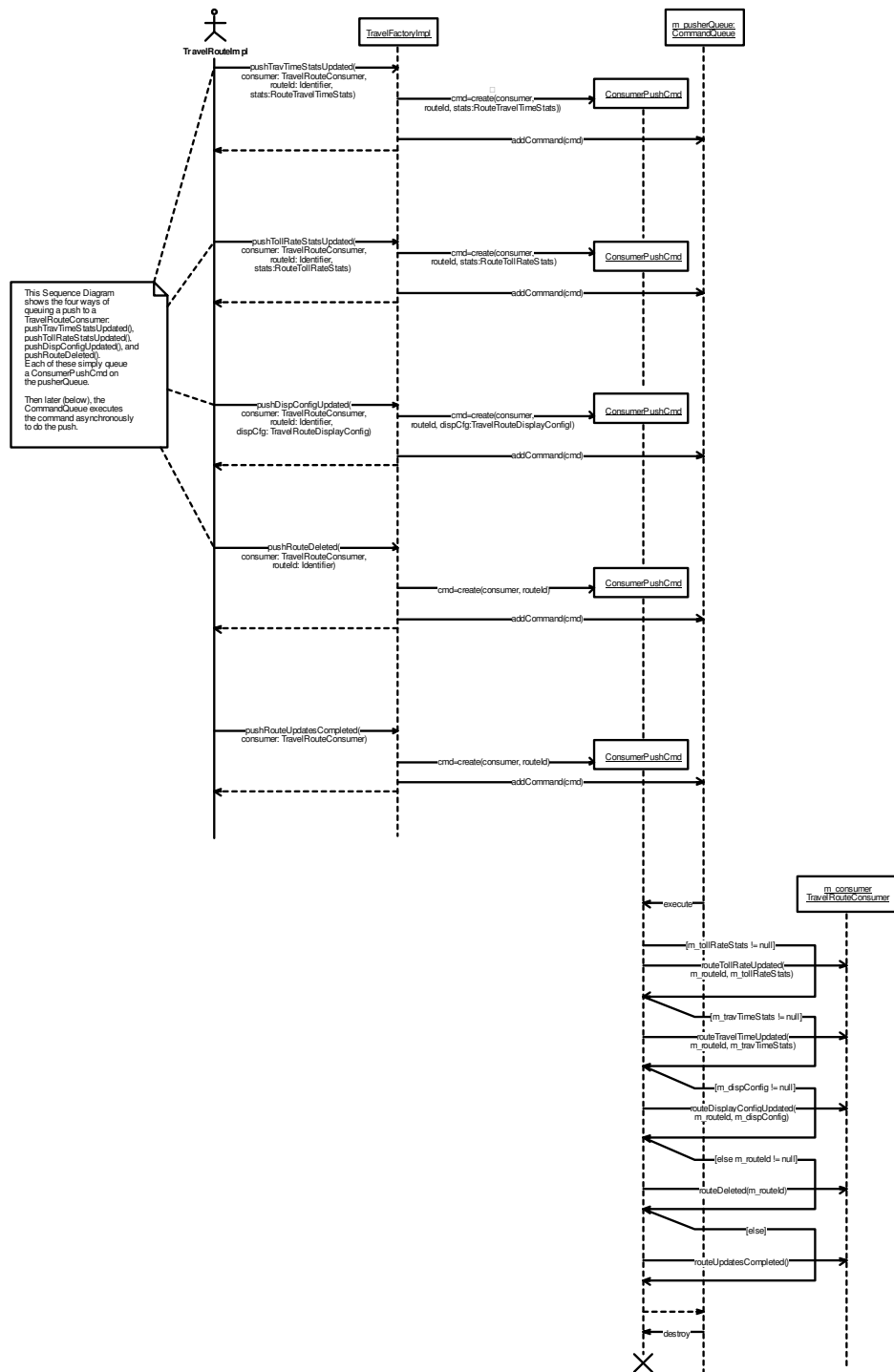


Figure 5-252. TravelRouteFactoryImpl:pushConsumerUpdate (Sequence Diagram)

5.20.2.6 TravelRouteFactoryImpl:sendUpdatesCompleted (Sequence Diagram)

This method informs TravelRouteConsumers that a batch of updates are complete, so that they can update their message, if they are holding their update actions in abeyance awaiting completion of the entire update activity. It is wise for a DMS to do this because typically many route updates are accomplished in close succession, as toll rates and link travel times are collected in bulk from those suppliers. In this way, waiting an extra few moments for all updates to come in, a DMS only has to communicate to a sign once per update cycle, instead of once for each route in its Traveler Information Message.

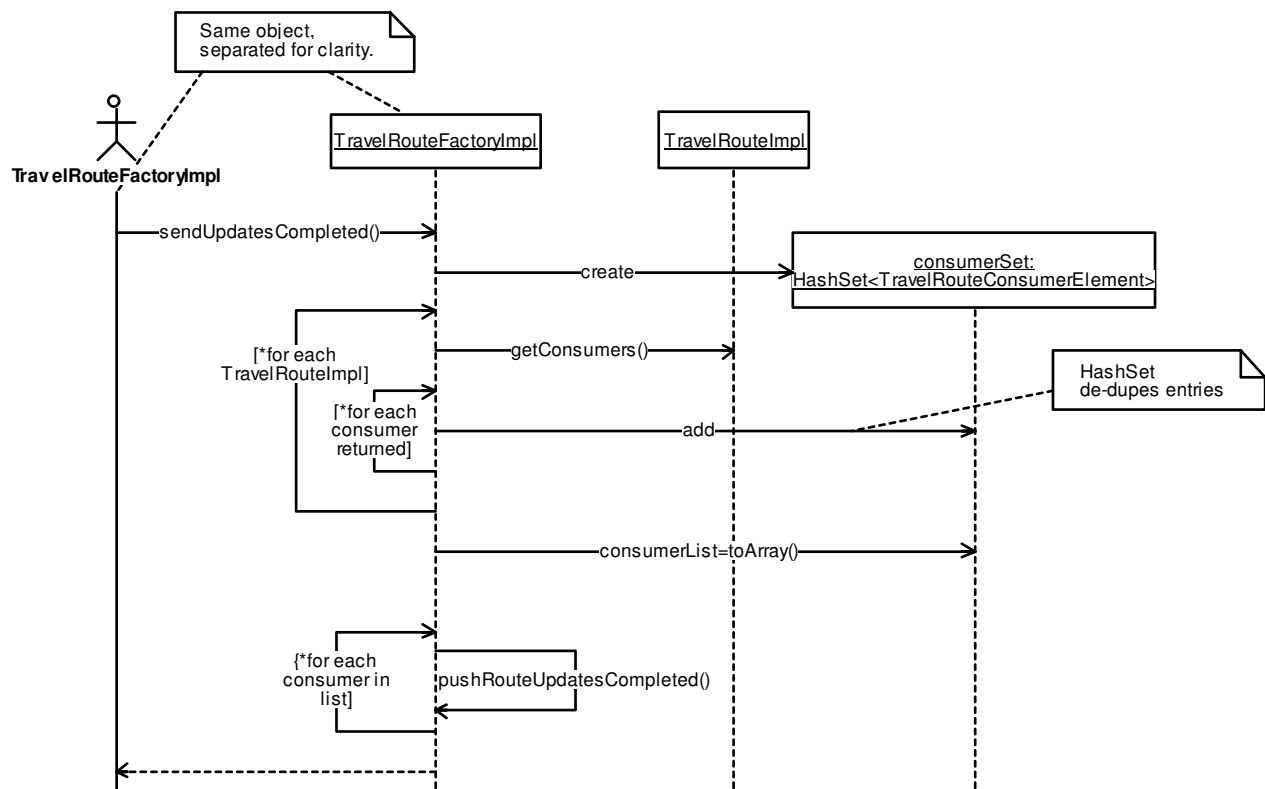


Figure 5-253. TravelRouteFactoryImpl:sendUpdatesCompleted (Sequence Diagram)

5.20.2.7 TravelRouteFactoryImpl:updateLinkData (Sequence Diagram)

This method is called by the INRIX Import Service to pass link travel times to the TravelRouteModule. The link data is first written to a temporary DB table so that control can quickly be returned to the caller, then the link data is processed asynchronously on the processing queue (which has just one thread). Each link's stats are updated, then each TravelRouteImpl is called to update the route travel time based on the new link travel times. An update for each consumer for each updated route is queued immediately, then when all routes have been updated, the updates completed message is queued to all consumers. See pushConsumerUpdate for consumer update processing.

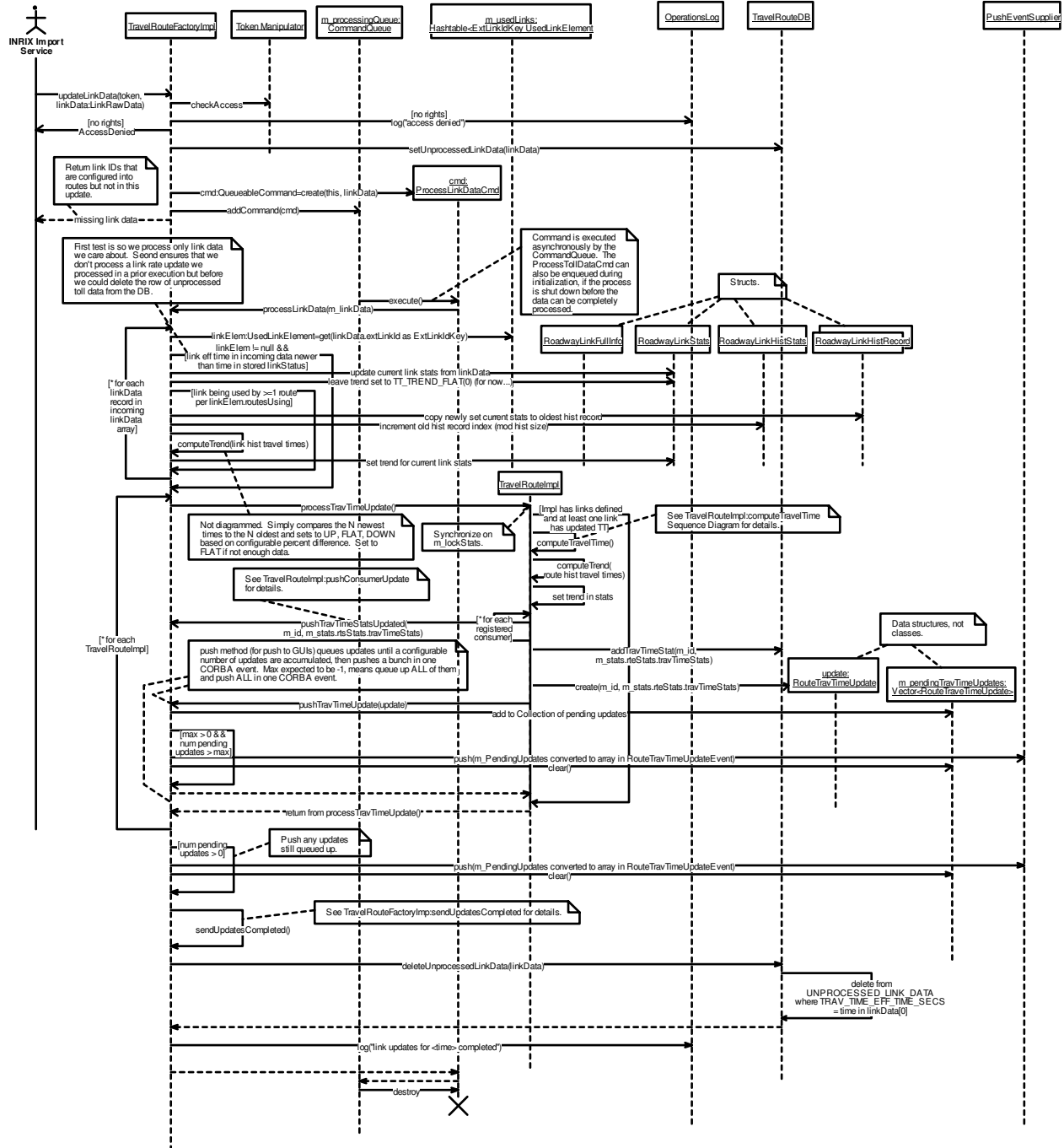
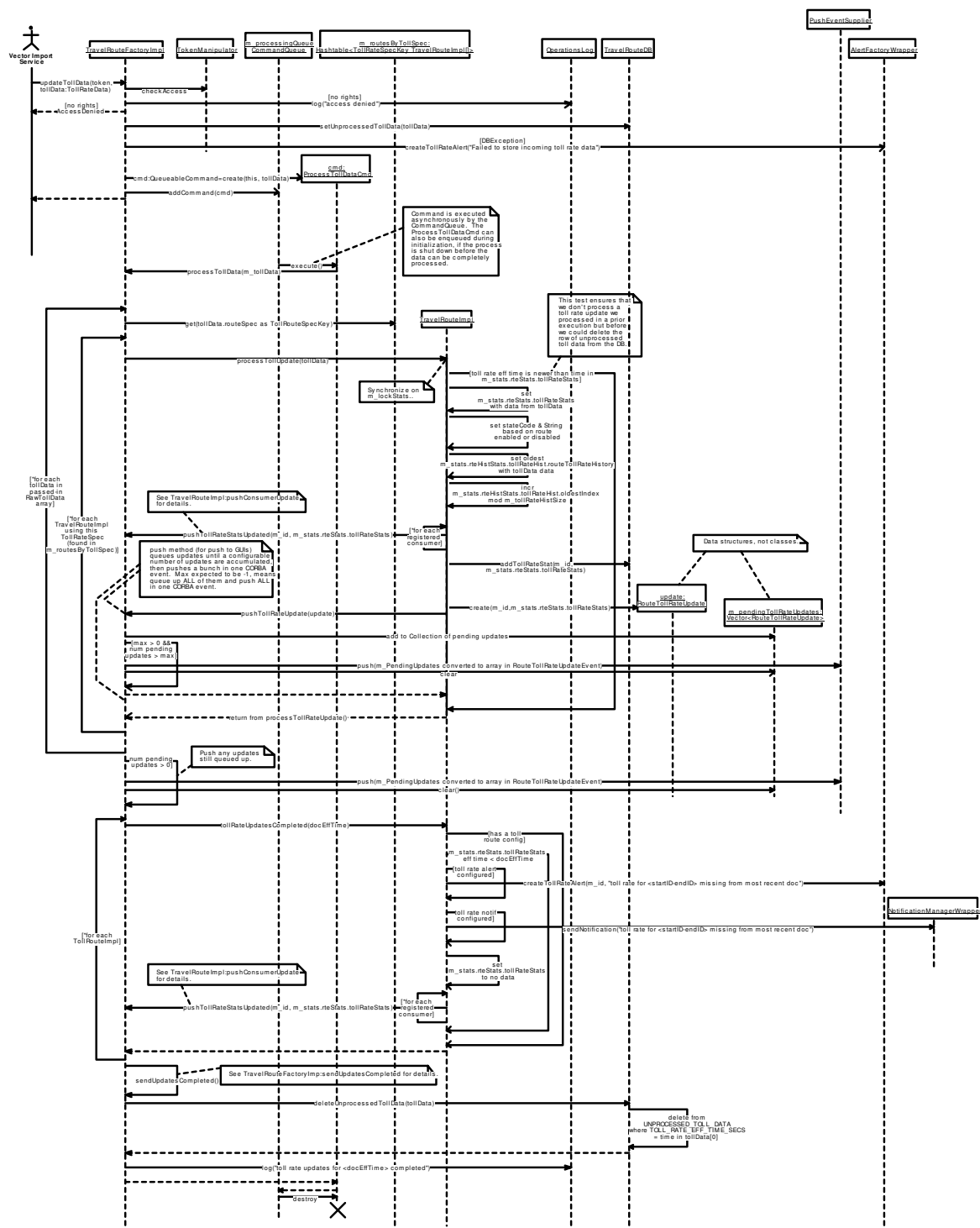


Figure 5-254. TravelRouteFactoryImpl:updateLinkData (Sequence Diagram)

5.20.2.8 TravelRouteFactoryImpl:updateTollRateData (Sequence Diagram)

This method is called by the VECTOR Import Service to pass toll rates to the TravelRouteModule. The toll rate data is first written to a temporary DB table so that control can quickly be returned to the caller, then the data is processed asynchronously on the processing queue (which has just one thread). For each rate passed in, each TravelRouteImpl using the rate is called to update its toll rate. An update for each consumer for each updated route is queued immediately, then when all routes have been updated, the updates completed message is queued to all consumers. See pushConsumerUpdate for consumer update processing.



5.20.2.9 TravelRouteImpl:addRemoveConsumer (Sequence Diagram)

This diagram shows two CORBA methods called on TravelRoute objects by TravelRouteConsumer objects. (A TravelRouteConsumer is, for example, a DMS which is using the route to feed an active Traveler Information Message.) A consumer (DMS) calls addConsumer when a Traveler Information Message goes active, and calls removeConsumer when the message is deactivated. In this way the only traffic flowing on the network is that which is necessary for a particular Traveler Information Message. When a consumer calls addConsumer(), the current stats are returned immediately so the consumer does not have to wait for the next update upon activating a message.

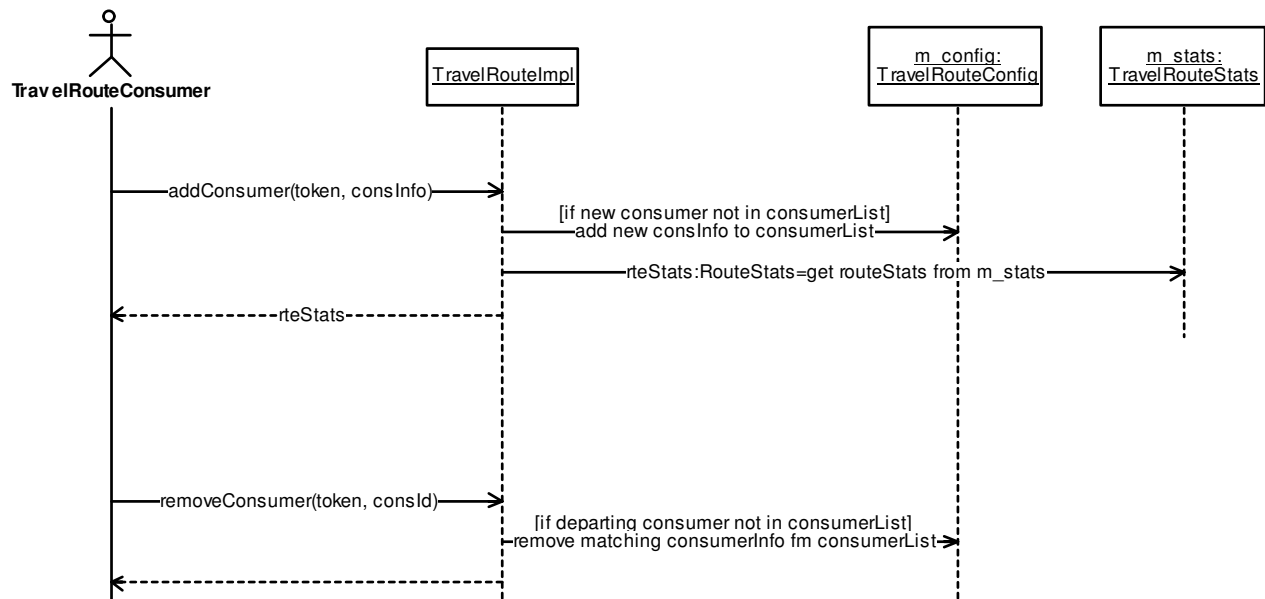


Figure 5-256. TravelRouteImpl:addRemoveConsumer (Sequence Diagram)

5.20.2.10 TravelRouteImpl:computeTravelTime (Sequence Diagram)

This helper method is called by a TravelRouteImpl on itself to compute its own travel time. It is called whenever it is known that the underlying link data has changed, or when the link configuration of the route has changed such that the route travel time calculation would be different. Basically the specified percentage of each link's travel times are added together. Errors are tracked and the state code and string are updated after the main loop. The trend is computed, and the stats record is added to the history before returning.



5.20.2.11 TravelRouteImpl:remove (Sequence Diagram)

This method is called by the GUI when a user wants to permanently remove a TravelRoute from the system. It first sets toll rate and travel time functions to disabled, and informs any connected consumers of that change, then it deletes itself from the database and the factory.

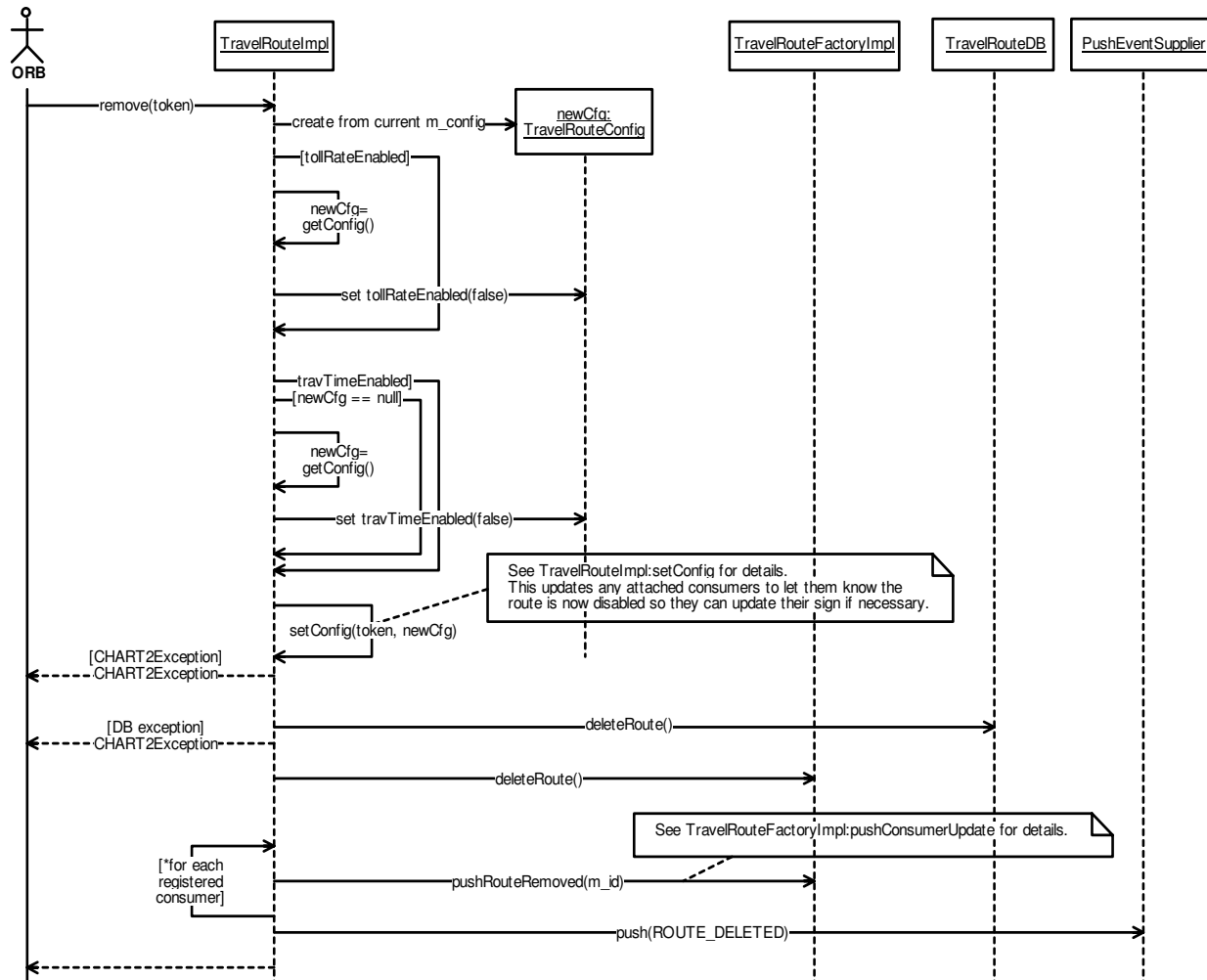


Figure 5-258. TravelRouteImpl:remove (Sequence Diagram)

5.20.2.12 TravelRouteImpl:setConfig (Sequence Diagram)

This method is called by the GUI when a user updates the configuration of a TravelRoute. This method checks to make sure certain things are not changed, like the route ID and the consumer list. Updates are made, and changes made are detected. If the travel time or toll rate processing is enabled or disabled, or if the travel time or toll rate configuration is changed, necessary updates are made to the current stats (e.g., recompute the travel time based on a new link configuration, new percentage(s) of links to include or minimum quality), and updates to the consumers and GUIs are pushed out.

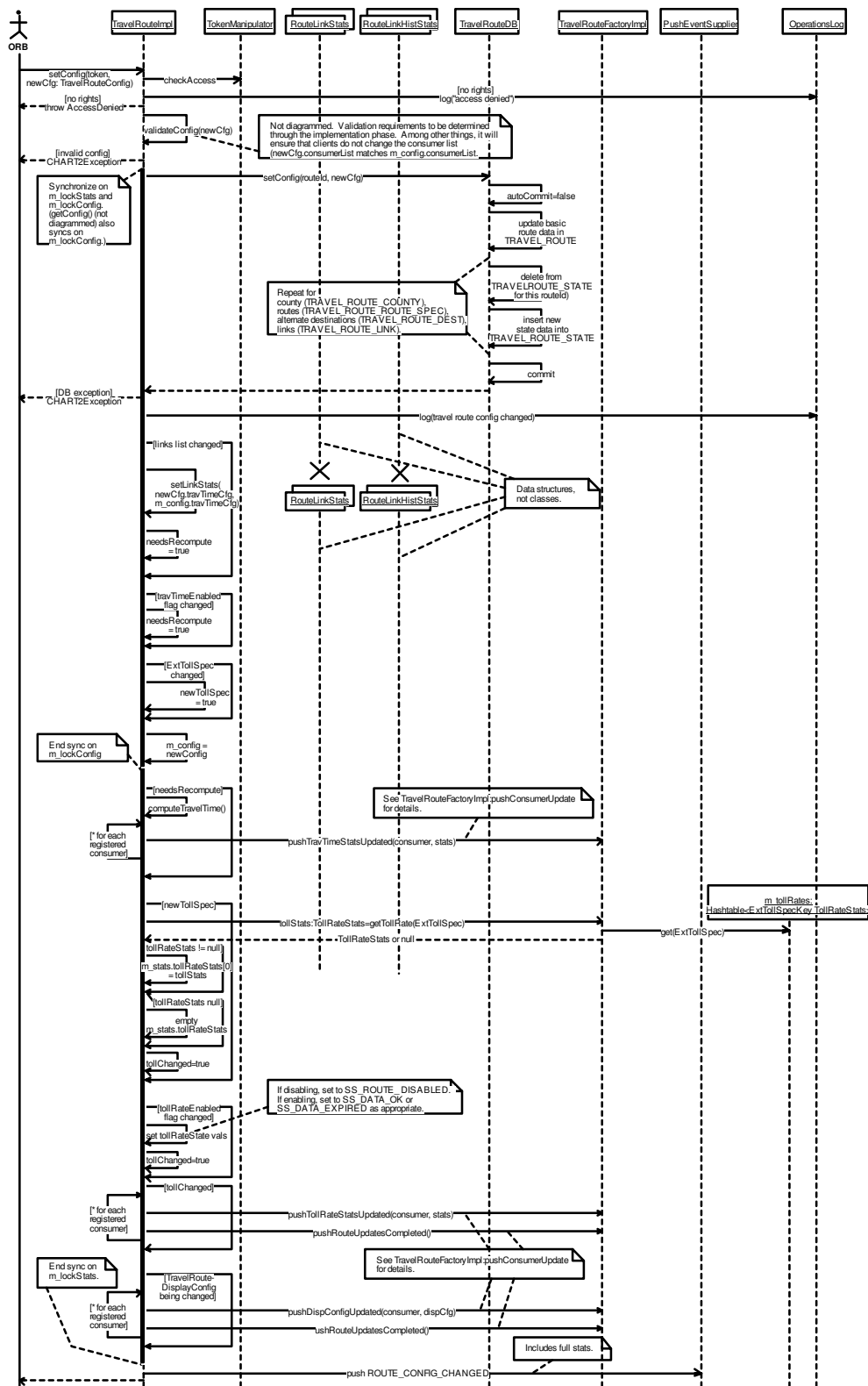


Figure 5-259. TravelRouteImpl:setConfig (Sequence Diagram)

5.20.2.13 TravelRouteImpl:setLinkStats (Sequence Diagram)

This helper method is called during a set configuration request when the route's link configuration has changed, to update the list of links which belong to the route and to attach the stats associated with those links, so that the new travel time then can be computed.

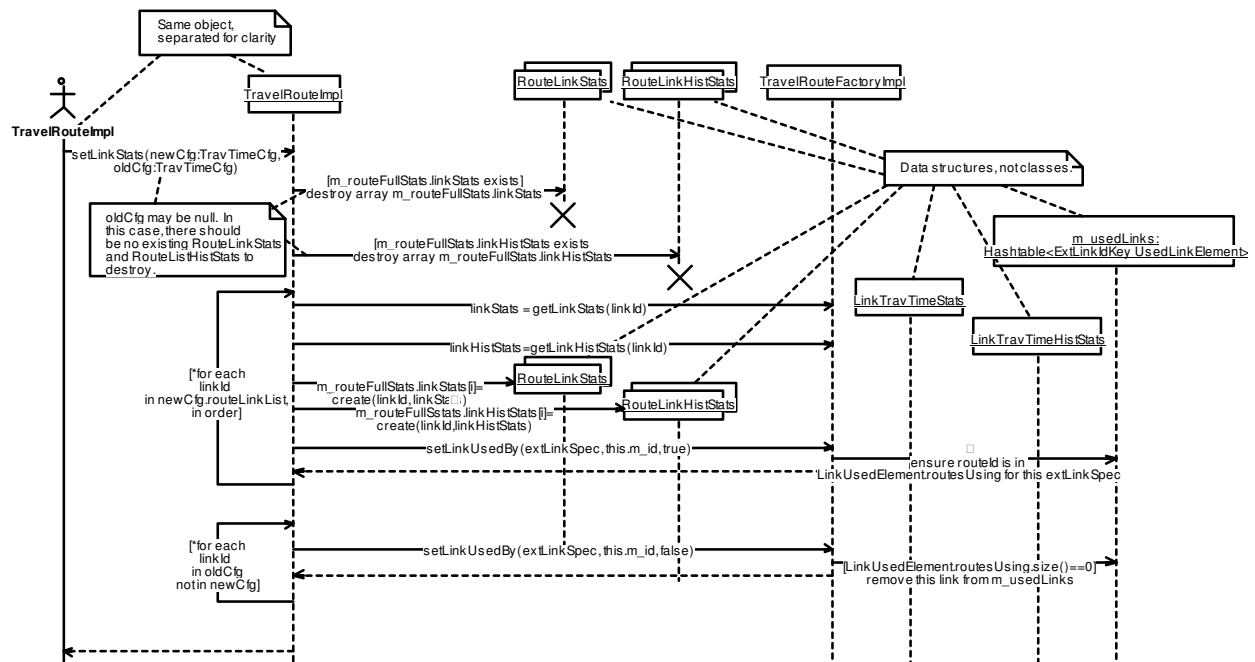


Figure 5-260. TravelRouteImpl:setLinkStats (Sequence Diagram)

5.20.2.14 TravelRouteImpl:setPartialConfig (Sequence Diagram)

This sequence diagram shows the quick implementation of two methods, `setTravelTimeConfig()` and `setTollRateConfig()`, which set just part of the entire `TravelRoute` configuration. These are implemented by attaching the new changed part of the configuration to the existing current configuration and then calling `setConfig()` on the `TravelRouteImpl` itself.

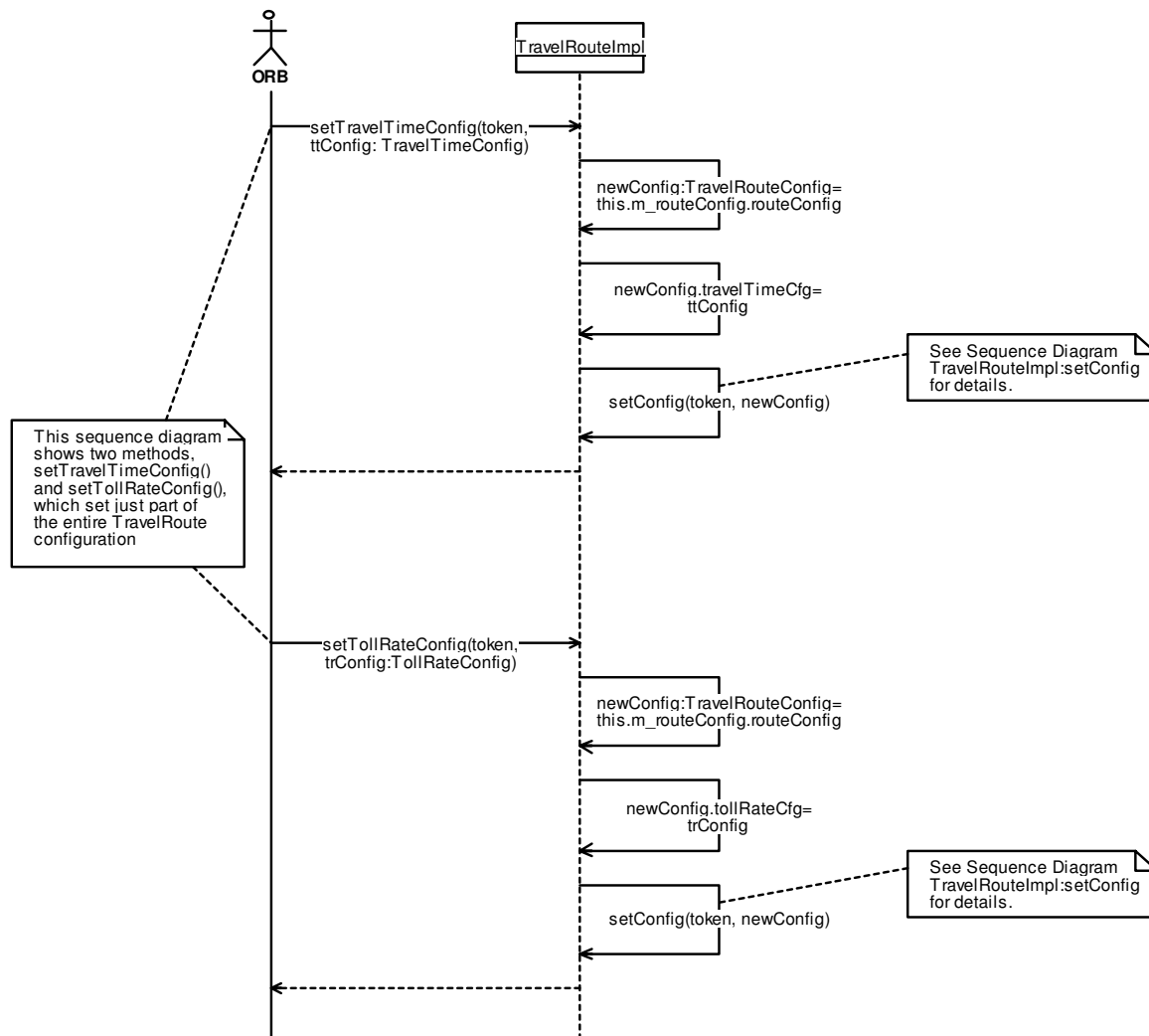


Figure 5-261. TravelRouteImpl:setPartialConfig (Sequence Diagram)

5.20.2.15 TravelRouteModule:initialize (Sequence Diagram)

This method shows initialization of the TravelRouteModule. This depersists the link data (stats) and travel routes and processes any queued up toll rate updates and/or link updates (which are not too old) from the VECTOR or INRIX import modules or which may be queued up but did not get a chance to be fully processed before the module was shut down. This method also starts a StalenessWatcher task which checks to make sure the route travel times do not get too stale (in the event that updates stop coming in from the INRIX import module). (Note: Toll Rates control their own expiration times directly, so the staleness watcher does not apply to toll rates).

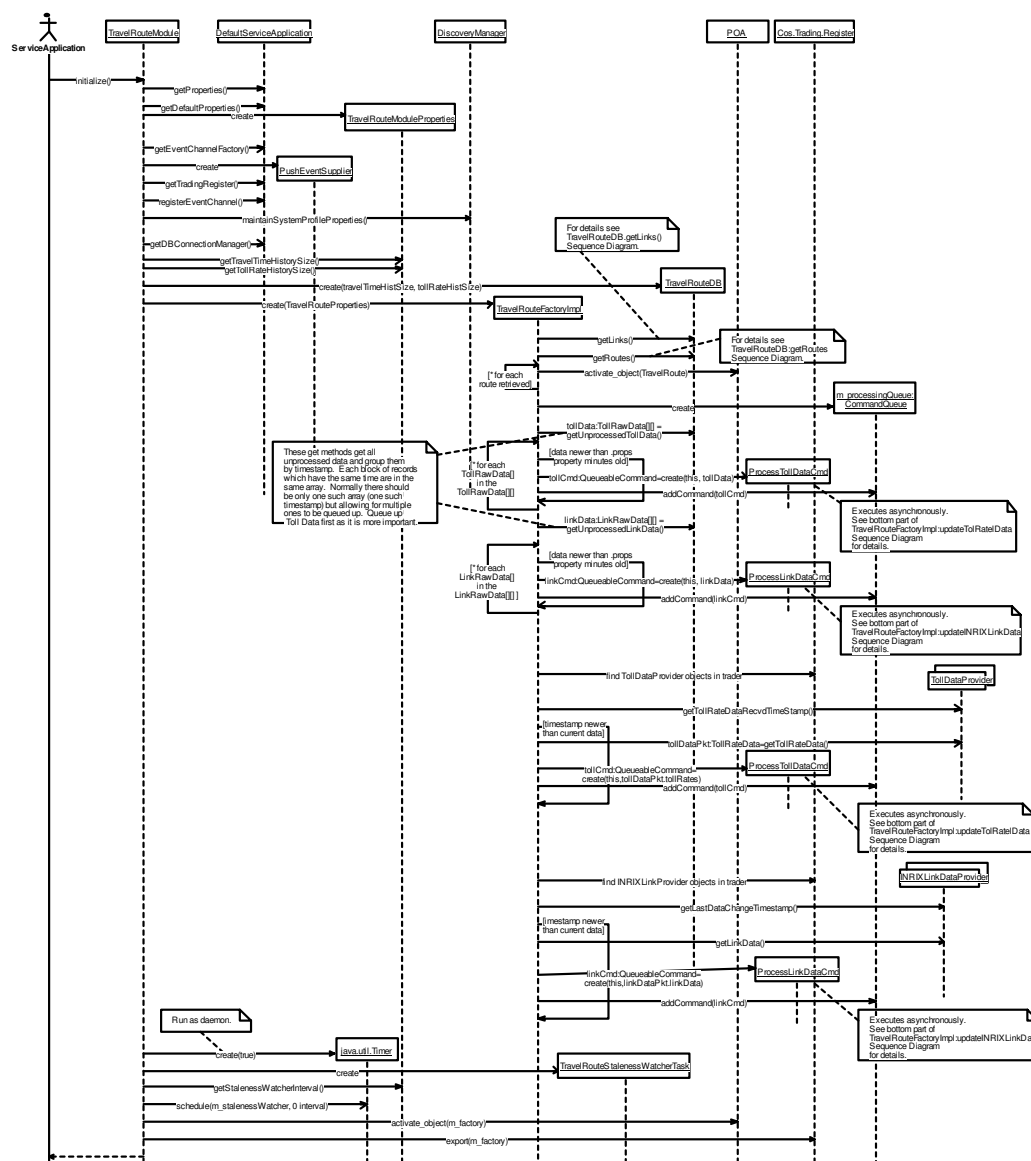


Figure 5-262. TravelRouteModule:initialize (Sequence Diagram)

5.20.2.16 TravelRouteModule:shutdown (Sequence Diagram)

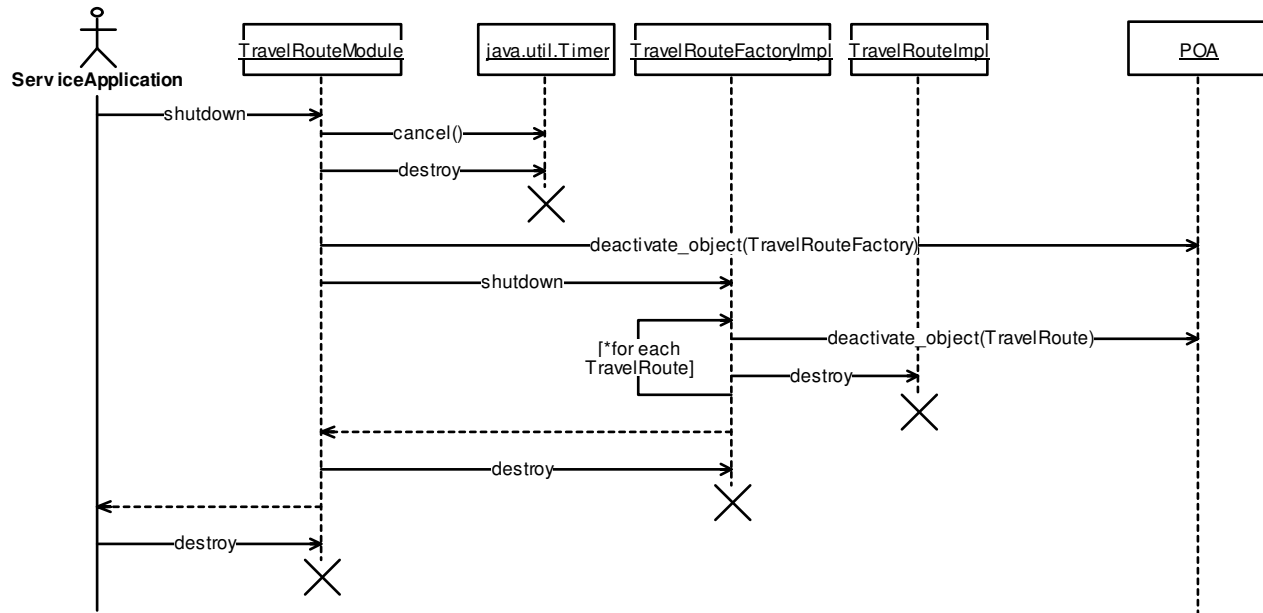


Figure 5-263. TravelRouteModule:shutdown (Sequence Diagram)

5.21 UserManagementmodulePkg

5.21.1 Classes

5.21.1.1 UserManagementClassDiagram (Class Diagram)

This class diagram depicts the CORBA IDL interface defined for user management in the CHART system. User management includes adding and deleting users from the system, modifying their roles, managing capabilities of roles, and management of system and user profile properties. The UserManager interface is largely an interface to the User Management database tables.

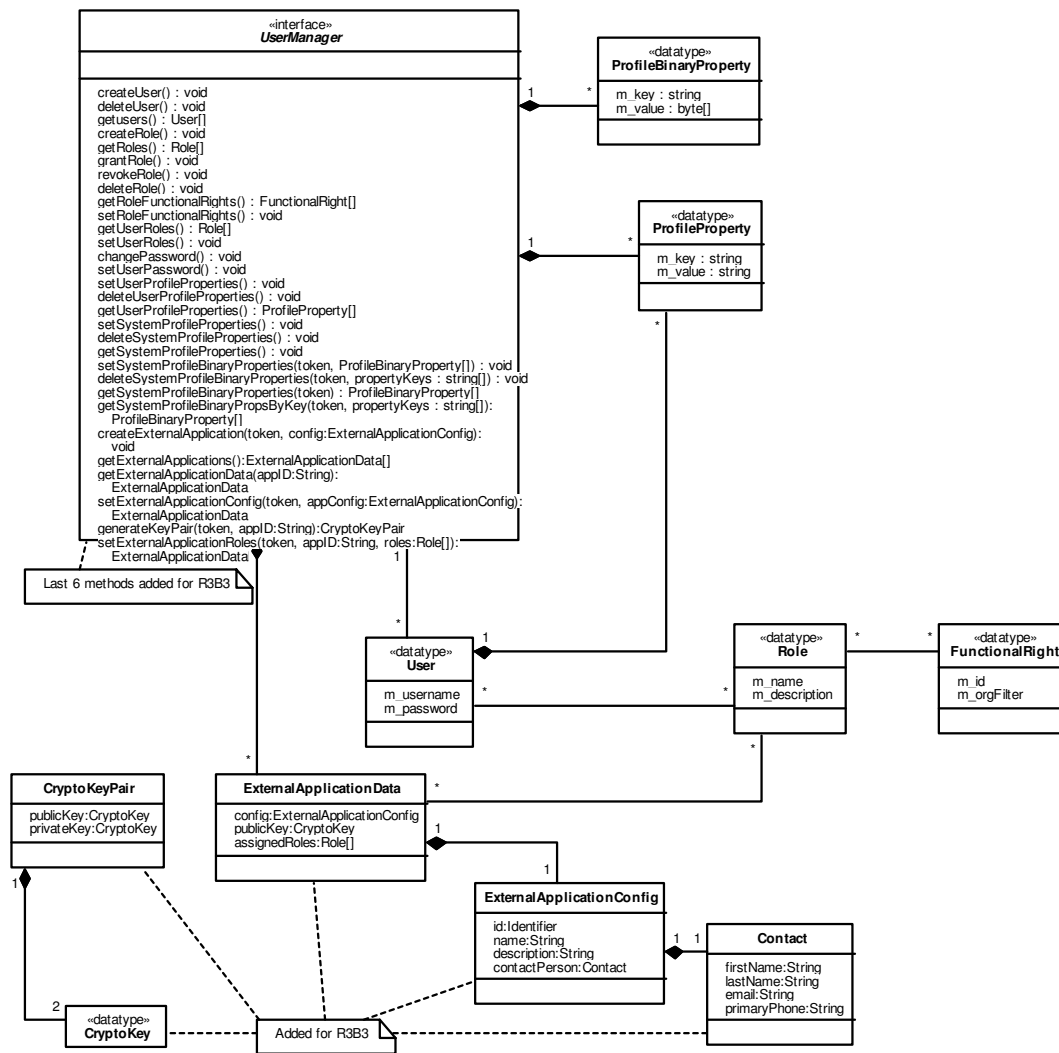


Figure 5-264. UserManagementClassDiagram (Class Diagram)

5.21.1.1.1 Contact (Class)

This class defines basic Contact data.

5.21.1.1.2 CryptoKey (Class)

This class represents a single key in a public/private key pair. It is an abstraction for an array of bytes.

5.21.1.1.3 CryptoKeyPair (Class)

This class represents a public/private key pair used by external client applications when communicating with CHART.

5.21.1.1.4 ExternalApplicationConfig (Class)

This class represents an exception thrown when an attempt is made to define a role which already exists.

5.21.1.1.5 ExternalApplicationData (Class)

This class represents the application data.

5.21.1.1.6 FunctionalRight (Class)

The FunctionalRights class represents the right to perform an action or set of actions. The functional right can be limited to apply to a single organization's shared resources. If the filter is not used, the functional right applies to all organization's shared resources.

5.21.1.1.7 ProfileBinaryProperty (Class)

This class represents a key value pair that can be used to store system properties in the system database. The key is a string and the value is binary data. One known use will be to store audio cue data to be played by the browser as part of the Alert Management capability introduced in R3B1.

5.21.1.1.8 ProfileProperty (Class)

This class represents a key value pair that can be used to store system properties in the system database. The key and the value are both strings.

5.21.1.1.9 Role (Class)

A Role is a collection of functional rights. A Role can be granted to a user, thus granting the user all functional rights contained within the role.

5.21.1.1.10 User (Class)

The User class represents a Chart II system user. In order to log into the Chart II system, a

user must be defined in the user database.

5.21.1.1.11 UserManager (Class)

The UserManager provides access to data dealing with user management. This includes users, roles, and functional rights. The UserManager is largely an interface to the User Management database tables.

5.22 UtilityPkg

5.22.1 Classes

5.22.1.1 UtilityClasses (Class Diagram)

This Class Diagram shows various utility classes that are used by various applications.

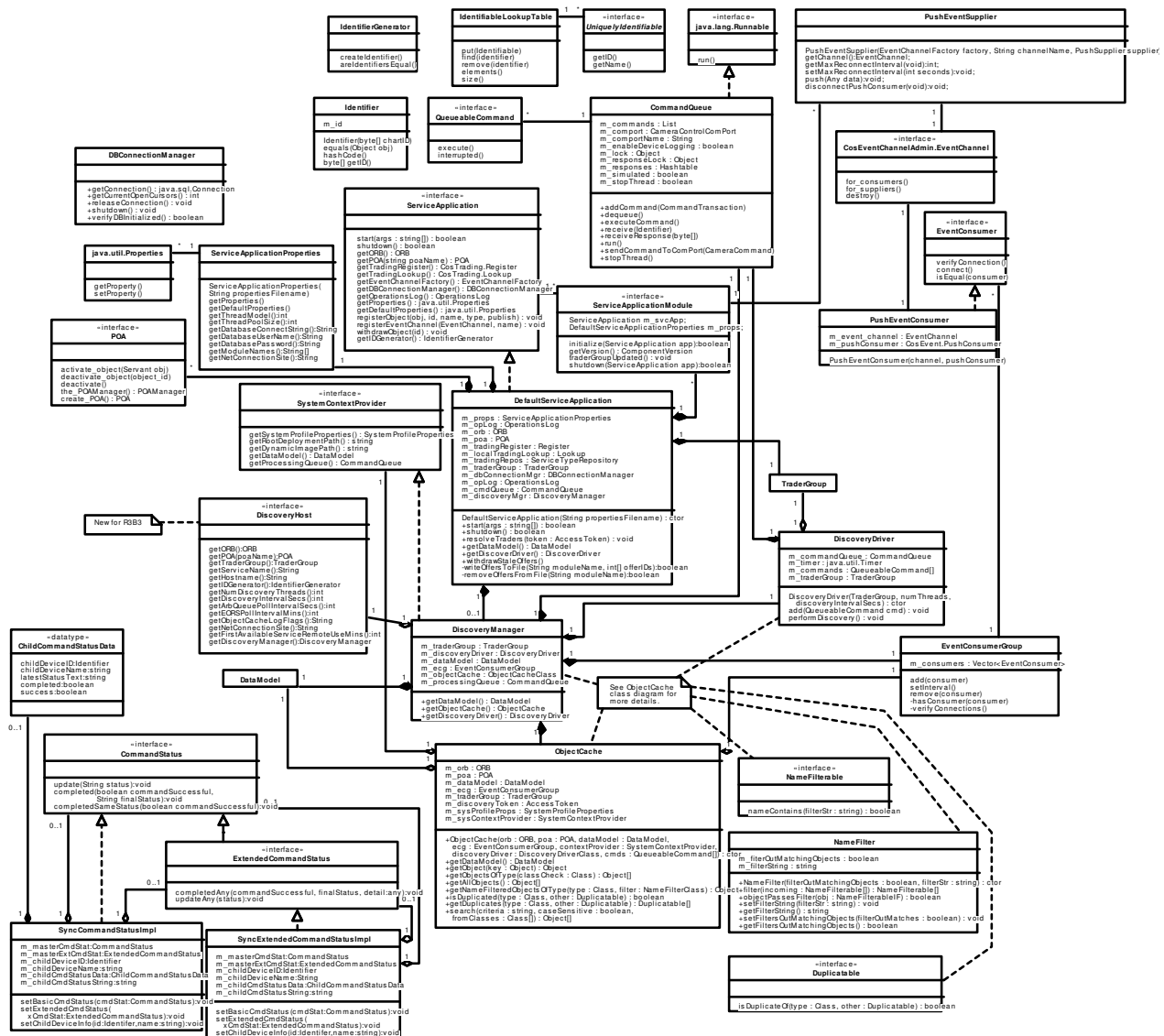


Figure 5-265. UtilityClasses (Class Diagram)

5.22.1.1.1 ChildCommandStatusData (Class)

This structure can be sent as the Any in an ExtendedCommandStatus back to clients. It contains information about a particular action which has taken place during processing of a long-running command fired off to one of any number of subsidiary "sub"-objects for completion of the primary task. This structure identifies the specific device for which the action has occurred, together with text and flags indicating the latest state of the command process on that device.

5.22.1.1.2 CommandQueue (Class)

The CommandQueue class provides a queue for QueueableCommand objects. The

CommandQueue has a thread that it uses to process each QueueableCommand in a first in first out order. As each command object is pulled off the queue by the CommandQueue's thread, the command object's execute method is called, at which time the command performs its intended task.

5.22.1.1.3 CommandStatus (Class)

The CommandStatus CORBA interface is used to allow a calling process to be notified of the progress of a long-running asynchronous operation. This is normally used when field communications are involved to complete a method call. The most common use is to allow a GUI to show the user the progress of an operation. It can also be used and watched by a server process when it needs to call on another server process to complete an operation. The long running operation typically calls back to the CommandStatus object periodically as the command is being executed, to provide in-progress status information, and it always makes a final call to the CommandStatus when the operation has completed. The final call to the CommandStatus from the long running operation indicates the success or failure of the command.

5.22.1.1.4 CosEventChannelAdmin.EventChannel (Class)

The event channel is a service that decouples the communication between suppliers and consumers of information.

5.22.1.1.5 DataModel (Class)

The data model class serves as a collection of objects. It provides an efficient lookup mechanism for locating any object, and methods which allow for the retrieval of all objects of a particular type. Additionally, this class provides the ability to attach observer objects which are notified when objects are added to or removed from the model. Objects may also notify the DataModel that they have been modified. The model will periodically notify all attached observers of the changes to objects in the model.

5.22.1.1.6 DBConnectionManager (Class)

This class implements a database connection manager that manages a pool of database connections. Any CHART II system thread requiring database access gets a database connection from the pool of connections maintained by this manager class. The connections are maintained in two separate lists namely, inUseList and freeList. The inUseList contains connections that have already been assigned to a thread. The freeList contains unassigned connections. This class assumes that an appropriate JDBC driver has been loaded either by using the "jdbc.drivers" system property or by loading it explicitly. The class has a monitor thread that is started by the constructor. This connection monitor thread periodically checks the inuseList to see if there are connections that are owned by dead threads and move such connections to the freeList. The connection monitor thread is started only if a non-zero value is specified for the monitoring time interval in the constructor.

5.22.1.1.7 DefaultServiceApplication (Class)

This class is the default implementation of the ServiceApplication interface. This class is passed a properties file during construction. This properties file contains configuration data used by this class to set the ORB concurrency model, determine which ORB services need to be available, provide database connectivity, etc. The properties file also contains the class names of service modules that should be served by the service application. During startup, the DefaultServiceApplication instantiates the service application module classes listed in the properties file and initializes each.

The DefaultServiceApplication maintains a file of offers that have been exported to the Trading Service. Each module must provide an implementation of the getOfferIDs method and be able to return the offer IDs for each object they have exported to the trader during their initialization. The DefaultServiceApplication stores all offer IDs in a file during its startup. Each module is expected to remove its offers from the trader during a shutdown. If the DefaultServiceApplication is not shutdown properly, it uses its offer ID file to clean-up old offers prior to initializing modules during its next start. This keeps multiple offers for the same object from being placed in the trader.

The DefaultServiceApplication also starts a DiscoveryManager. (If no modules add discovery QueueableCommand objects to the DiscoveryManager's DiscoveryDriver, discovery runs, but does nothing, so incurs virtually no cost.)

5.22.1.1.8 DiscoveryDriver (Class)

This class drives the periodic discovery of objects from other services within the CHART system. Other objects in the system that need access to other service's objects add their own QueueableCommand to the DiscoveryDriver. Each time discovery is performed, the discovery driver uses a command queue to execute all queueable commands that have been added in a separate thread of execution. The commands are added to the command queue immediately upon execution, and then executed in serial fashion via the command queue until all commands have executed. The frequency of discovery is controlled by a property. Discovery occurs more frequently immediately after service startup, to more quickly discover objects from other services which may also be starting up at more or less the same time. The DiscoveryDriver can be configured to have multiple threads to allow concurrent discovery of different objects.

5.22.1.1.9 DiscoveryHost (Class)

This interface defines the methods that the DiscoveryManager relies on. It must be implemented by any class that will create a DiscoveryManager.

5.22.1.1.10DiscoveryManager (Class)

This SystemContextProvider interface defines some of the functionality required by a class which provides discovery services for CHART services. It is used by both the CHART GUI and the CHART backend services. A class which implements this interface must provide "get" accessor methods for the system profile properties, the data model, and the

main processing queue for a service, for instance. It also provides access to the root deployment path and dynamic image path, which is used only by the CHART GUI. For the CHART GUI, this interface is known to be implemented by the MainServlet; for the back end CHART services, this interface is known to be implemented by the Discovery Manager.

5.22.1.1.11 Duplicatable (Class)

This java interface is implemented by classes which have sense of being "duplicated" within the CHART system. This allows the ObjectCache to search for duplicates of any Duplicatable object. This is different from "equals()" or "compareTo()". To cite two examples: Alerts within CHART are duplicates if they refer to the same objects within CHART (but do not have the same Alert ID, which is more closely associated with "equals()"). Traffic Events within CHART are duplicates if they have the same location (but do not have the same Traffic Event ID).

5.22.1.1.12 EventConsumer (Class)

This interface provides the methods which any EventConsumer object that would like to be managed in an EventConsumerGroup must implement.

5.22.1.1.13 EventConsumerGroup (Class)

This class represents a collection of event consumers which will be monitored to verify that they do not lose their connection to the CORBA event service. The class will periodically ask each consumer to verify its connection to the event channel on which it is dependant to receive events.

5.22.1.1.14 ExtendedCommandStatus (Class)

The ExtendedCommandStatus CORBA interface is used to allow a calling process to be notified of the progress of a long-running asynchronous operation. This interface extends the basic CommandStatus interface by allowing additional information to be passed in in a CORBA "Any" object. The "Any" can be configured to hold (as the name would suggest) any type of information. The information defined in the any varies based on the particular command being called. The most common use is to allow a GUI to show the user the progress of an operation. It can also be used and watched by a server process when it needs to call on another server process to complete an operation. The long running operation typically calls back to the ExtendedCommandStatus object periodically as the command is being executed, to provide in-progress status information, and it always makes a final call to the CommandStatus when the operation has completed. The final call to the ExtendedCommandStatus from the long running operation indicates the success or failure of the command.

5.22.1.1.15 IdentifiableLookupTable (Class)

This class uses a hash table implementation to store Identifiable objects for fast lookups.

5.22.1.1.16Identifier (Class)

Wrapper class for a CHART2 identifier byte sequence. This class will be used to add identifiable objects to hash tables and perform subsequent lookup operations.

5.22.1.1.17IdentifierGenerator (Class)

This class is used to create and manipulate identifiers which are to be used in Identifiable objects.

5.22.1.1.18java.lang.Runnable (Class)

This interface allows the run method to be called from another thread using Java's threading mechanism.

5.22.1.1.19java.util.Properties (Class)

The Properties class represents a persistent set of properties. The Properties can be saved to a stream or loaded from a stream. Each key and its corresponding value in the property list is a string. A property list can contain another property list as its "defaults"; this second property list is searched if the property key is not found in the original property list.

5.22.1.1.20NameFilter (Class)

This class defines a filter by which a NameFilterable object can be selected from the ObjectCache. It provides a string to search for, and a flag to indicate whether the desired result is those object which match the filter, or those which do not.

5.22.1.1.21NameFilterable (Class)

This java interface is implemented by classes which can be filter by name within the ObjectCache. A NameFilter object is passed into the ObjectCache to select NameFilterable objects in the cache.

5.22.1.1.22ObjectCache (Class)

The ObjectCache is a wrapper for the DataModel. It provides access to DataModel methods to find objects in the data model, delegating those methods to the DataModel itself. It also provides additional methods of finding name filtered objects and discovering "duplicate" objects (as defined by an isDuplicateOf() method of the Duplicatable interface).

5.22.1.1.23POA (Class)

This interface represents the portable object adapter used to activate and deactivate servant objects.

5.22.1.1.24PushEventConsumer (Class)

This class is a utility class which will be responsible for connecting a consumer implementation to an event channel, and maintaining that connection. When the

verifyConnection method is called, this object will determine if the channel has been lost and will attempt to re-connect to the channel if it has.

5.22.1.1.25PushEventSupplier (Class)

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

5.22.1.1.26QueueableCommand (Class)

A QueueableCommand is an interface used to represent a command that can be placed on a CommandQueue for asynchronous execution. Derived classes implement the execute method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a CommandQueue. The CommandQueue driver calls the execute method to execute a command in the queue and a call to the interrupted method is made when a CommandQueue is shut down.

5.22.1.1.27ServiceApplication (Class)

This interface is implemented by objects that can provide the basic services needed by a ChartII service application. These services include providing access to basic CORBA objects that are needed by service applications, such as the ORB, POA, Trader, and Event Service.

5.22.1.1.28ServiceApplicationModule (Class)

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

5.22.1.1.29ServiceApplicationProperties (Class)

This class provides methods which allow the DefaultServiceApplication to access the necessary properties from the java properties configuration file. It also provides a default properties file which can be retrieved by anyone holding a ServiceApplication interface reference. This gives each installed service module the opportunity to load default values

before retrieving property values from the properties file.

5.22.1.1.30 SyncCommandStatusImpl (Class)

This is an implementation of CommandStatus which can be used by server-side processes which need to kick off and check results of multiple long-running commands. The SyncCommandStatusImpl can notify a MuxWaitSem object when the CommandStatus completed() call is made (meaning the long-running command has completed). (The MuxWaitSem can be waited on until all such SyncCommandStatusImpl objects have completed.) Additionally, new in R2B3, the SyncCommandStatusImpl has the facility to take a "master" CommandStatus or ExtendedCommandStatus (expected to be held by client code) which can receive results from the various "child" CommandStatus objects. If the master is a simple CommandStatus, the results are sent "inline" as additional text messages in update() calls, for unstructured, unsorted display to the user. If the master is an ExtendedCommandStatus, the results are sent in a ChildCommandStatusData object, for more organized display to the user.

5.22.1.1.31 SyncExtendedCommandStatusImpl (Class)

This is an implementation of ExtendedCommandStatus which can be used by server-side processes which need to kick off and check results of multiple long-running commands. The SyncExtendedCommandStatusImpl can notify a MuxWaitSem object when the CommandStatus completed() call is made (meaning the long-running command has completed). (The MuxWaitSem can be waited on until all such SyncExtendedCommandStatusImpl objects have completed.) Additionally, new in R2B3, the SyncExtendedCommandStatusObject has the facility to take a "master" CommandStatus or ExtendedCommandStatus (expected to be held by client code) which can receive results from the various "child" CommandStatus objects. If the master is a simple CommandStatus, the results are sent "inline" as additional text messages in update() calls, for unstructured, unsorted display to the user. If the master is an ExtendedCommandStatus, the results are sent in a ChildCommandStatusData object, for more organized display to the user.

5.22.1.1.32 SystemContextProvider (Class)

Interface which provides operations for access to few key objects, and is implemented by the DiscoveryManager

5.22.1.1.33 TraderGroup (Class)

This class provides a facade for trader lookups that allows application level code to be unaware of the number of CORBA trading services that the application is using or the details of the linkage between those services.

5.22.1.1.34 UniquelyIdentifiable (Class)

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

This Class Diagram shows various utility classes related to log entries that are used by GUI and servers.



This class is designed to contain a collection of comparable objects. All of the objects added to this collection must be of the same concrete type. Each element in the collection has an associated counter which tracks how many times this element has been added. It is then possible to get only the elements which have been added to the collection n times

where n is a positive integer value. This class is very useful for creating GUI menu's for multiple objects as it allows all objects to insert their menu items and then allows the user to get only those items which all objects inserted.

5.22.1.2.2 CachedLogEntry (Class)

This class represents a reference-counting object stored in a memory-efficient LogEntryCache. The object of this class encapsulates the stored log entry and adds a reference count.

5.22.1.2.3 CorbaUtilities (Class)

This class is a collection of static CORBA utility methods that can be used by both server and GUI for CORBA Trader service transactions.

5.22.1.2.4 DatabaseLogger (Class)

This class represents a generic database logger which can be used to log and retrieve information from the database. This class also provides a mechanism for the user to filter and retrieve logs that meet a specific criteria.

5.22.1.2.5 DBUtility (Class)

This class contains methods that allow interaction with the database.

5.22.1.2.6 DMSHardwarePage (Class)

This class holds data that specifies the layout of one page of a DMS message on the actual DMS hardware. A two dimensional array that is the same size as the sign's display (rows and columns) specifies the character displayed in each cell, including blank if the cell has no character. This format maps well to the way DMS protocols return the current message being displayed in a status query. This class can then be passed to a MultiConverter object to convert the message into MULTI format.

5.22.1.2.7 FunctionalRightType (Class)

This class acts as an enumeration that lists the types of functional rights possible in the CHART2 system. It contains a static member for each possible functional right.

5.22.1.2.8 GeoAreaUtil (Class)

This class contains static methods used for searching GeoAreas for specific Lat/Lons. The actual search of a GeoArea is done by converting a chart GeoArea to a java.awt.Polygon then using that object's contains(lat,lon) method to make the determination. Several helper methods front the Polygon based methods for flexibility.

5.22.1.2.9 Log (Class)

Singleton log object to allow applications to easily create and utilize a LogFile object for

system trace messages.

5.22.1.2.10LogEntry (Class)

This class represents a typical log entry that is stored in the database. This can be a general Communications Log entry or it can be a historical entry for a Traffic Event. Some Traffic Event actions (opening, closing, etc.) are logged in the Communications Log as well as in the history of the specific Traffic Event.

5.22.1.2.11LogEntryCache (Class)

The LogEntryCache caches log entries returned from a database query which are in excess of the requestor-specified maximum number of entries to return at one time. The LogIterator stores references to the LogEntry objects thus cached, and requests additional objects as needed. The LogEntryCache uses reference counting to prevent storing duplicate copies of LogEntry objects, and it deletes LogEntry objects when they are no longer needed.

5.22.1.2.12LogFile (Class)

This class creates a flat file for writing system trace log messages and purges them at user specified interval. The log files created by this class are used for system debugging and maintenance only and are not to be confused with the system operations log which is modeled by the OperationsLog class.

5.22.1.2.13LogFilter (Class)

This class is used to specify the criteria to be used when getting entries from the Communications Log. The caller would create an object of this type specifying the criteria that each log entry must match in order to be returned.

5.22.1.2.14LogIterator (Class)

This class represents an iterator to iterate through a collection of log entries. If a retrieval request results in more data than is reasonable to transmit all at once, one clump of entries is returned at first, together with a LogIterator from which additional data can be requested, repeatedly, until all entries are returned or the user cancels the operation.

5.22.1.2.15LogIteratorImpl (Class)

The LogIteratorImpl implements the LogIterator interface; that is, it does the actual work which clients can request via the LogIterator interface. The LogIteratorImpl stores data relating to cached LogEvents for a single retrieval request, and implements the client request to get additional clumps of data pertaining to that request.

5.22.1.2.16MultiConverter (Class)

This class provides methods which perform conversions between the DMS MULTI mark-up language and plain text. It also provides a method which will parse a MULTI message

and inform a MultiParseListener of elements found in the message.

5.22.1.2.17MultiFormatter (Class)

This interface must be implemented by classes which convert plain text DMS messages to MULTI formatted messages.

5.22.1.2.18MultiParseListener (Class)

A MultiParseListener works in conjunction with the MultiConverter to allow an implementing class to be notified as parsing of a MULTI message occurs. An exemplary use of a MultiParseListener would be the MessageView window which will need to have the MULTI message parsed in order to display it as a pixmap.

5.22.1.2.19ObjectCache (Class)

The ObjectCache is a wrapper for the DataModel. It provides access to DataModel methods to find objects in the data model, delegating those methods to the DataModel itself. It also provides additional methods of finding name filtered objects and discovering "duplicate" objects (as defined by an isDuplicateOf() method of the Duplicatable interface).

5.22.1.2.20ObjectLocator (Class)

This class is used to provide access to proxy objects stored in the CHART object cache (which have been discovered by the DiscoveryDriver tasks)

5.22.1.2.21OperationsLog (Class)

This class provides the functionality to add a log entry to the Chart II operations log. At the time of instantiation of this class, it creates a queue for log entries. When a user of this class provides a message to be logged, it creates a time-stamped OpLogMessage object and adds this object to the OpLogQueue. Once queued, the messages are written to the database by the queue driver thread in the order they were queued.

5.22.1.2.22OpLogMessage (Class)

This class holds data for a message to be stored in the system's Operations Log.

5.22.1.2.23OpLogQueue (Class)

This class is a queue for messages that are to be put into the system's Operations Log. Messages added to the queue can be removed in FIFO order.

5.22.1.2.24ProxyObject (Class)

This class is a base class for many types of proxy objects store in the CHART object cache (which have been discovered by the DiscoveryDriver tasks), used to provide a standard set of access methods for the proxy objects.

5.22.1.2.25TokenManipulator (Class)

This class contains all functionality required for user rights in the system. It is the only code in the system which knows how to create, modify and check a user's functional rights. It encapsulates the contents of an octet sequence which will be passed to every secure method. Secure methods should call the checkAccess method to validate the user. Client processes should use the check access method to verify access and optimize to reduce the size of the sequence to only those rights which are necessary to invoke the secure method. The token contains the following information. Token version, Token ID, Token Time Stamp, Username, Op Center ID, Op Center IOR, functional rights

5.22.1.2.26TraderGroup (Class)

This class provides a facade for trader lookups that allows application level code to be unaware of the number of CORBA trading services that the application is using or the details of the linkage between those services.

5.22.1.2.27TravelTimeRange (Class)

This class is a utility that can convert a travel time into a travel time range. It is constructed using the travel time range definitions as specified by a JSON string stored in the system profile. The convertToRange method can then be called get the low and high range values for a specific travel time.

5.22.1.2.28TravelTimeRangeDef (Class)

This class holds data for a travel time range definition. It has methods that can convert this object to a JSON object for persistence in the system profile, and to allow its data to be loaded from a JSON object when depersisting from the system profile.

5.22.1.2.29TravelTimeScheduleUtil (Class)

This class provides utility methods related to travel time schedules.

5.22.2.1 TravelTimeRange:constructor (Sequence Diagram)

```
sequenceDiagram
    actor System
    participant TravelTimeRange
    participant JSONValue
    participant JSONArray
    participant JSONObject
    participant Number
    participant TravelTimeRangeDef as TravelTimeRangeDef[]

    System->>TravelTimeRange: create(json:String)
    activate TravelTimeRange
    TravelTimeRange->>JSONValue: parse(String)
    JSONValue-->>TravelTimeRange: JSONArray
    deactivate JSONValue
    TravelTimeRange->>JSONArray: toArray()
    JSONArray-->>TravelTimeRange: JSONObject[]
    deactivate JSONArray
    TravelTimeRange->>TravelTimeRangeDef: create
    activate TravelTimeRangeDef
    TravelTimeRange->>JSONObject: get("travelTime")
    JSONObject-->>TravelTimeRange: Number
    deactivate JSONObject
    TravelTimeRange->>Number: intValue()
    Number-->>TravelTimeRange: int
    deactivate Number
    Note over Number, TravelTimeRangeDef: repeat these calls for "subtractAmt" and "addAmt" fields.
    TravelTimeRange->>TravelTimeRangeDef: create(travelTime, subtractAmt, addAmt)
    activate TravelTimeRangeDef
    TravelTimeRange->>TravelTimeRangeDef: store TravelTimeRangeDef
    deactivate TravelTimeRangeDef
    TravelTimeRange-->>System: TravelTimeRange
    deactivate TravelTimeRange
```

CHART R3B3 Detailed Design

5.23 UtilityPkg.wrappers

5.23.1 Classes

5.23.1.1 WrappersCD (Class Diagram)

This class diagram shows how wrappers work within CHART. Several types of high level object within the CHART system (typically "Factory" or "Manager" objects) exist on many servers. These objects are often referenced by numerous other modules within the system. Some effort is involved in gaining and maintaining references to these objects. All of that logic is hidden behind a "wrapper" class, so that a client can use the wrapper without worrying about gaining and maintaining references to the desired objects. Two types of wrappers are provided: A "primary-first" wrapper, which keeps and returns a reference to a "preferred" instance of the object whenever possible, and a "first-available" wrapper, which returns any instance of the object (but still prefers a local instance of the object whenever available).

This class delegates all of its method calls to the system AlertFactory using its currently known good reference to an AlertFactory. If the current reference returns a CORBA failure in the delegated call, this class automatically switches to another reference. When there are no good references (as is true the first time the object is used), this class issues a trader query to (re)discover the published Alert Factory objects in the system. During a method call, the trader will be queried at most one time and under normal circumstances, not at all.

5.23.1.1.3 Dictionary (Class)

The Dictionary IDL interface provides functionality to add, delete and check for words that are approved or banned from being used in CHART2 messaging devices such as HARs and DMSs. It also provides functionality to manage pronunciations.

5.23.1.1.4 DictionaryWrapper (Class)

This singleton class provides a wrapper for the system dictionary that provides automatic location of the dictionary and automatic re-discovery should the dictionary reference return an error. This class also allows for built-in fault tolerance by automatically failing over to a "working" dictionary without the user of this class being aware that this being done. In addition, this class defers the discovery of the Dictionary until its first use, thus eliminating a start-up dependency for modules that use the dictionary.

This class delegates all of its method calls to the system dictionary using its currently known good reference to the system dictionary. If the current reference returns a CORBA failure in the delegated call, this class automatically switches to another reference. When there are no good references (as is true the first time the object is used), this class issues a trader query to (re)discover the published Dictionary objects in the system. During a method call, the trader will be queried at most one time and under normal circumstances (other than the first use) the trader will not be queried at all.

5.23.1.1.5 FirstAvailableOfferWrapper (Class)

This class is a generic wrapper that provides the ability to find the first available reference to a service that may have multiple instances within the system.

5.23.1.1.6 NotificationManager (Class)

Interface whose implementation is used to manage notification messages, retrieve notification recipients (groups and individuals) and query notification status records.

5.23.1.1.7 NotificationManagerWrapper (Class)

This singleton class presents the same interface as the NotificationManager, but uses a FirstAvailableOfferWrapper to provide fault tolerant access to the methods.

5.23.1.1.8 OfferWrapper (Class)

An OfferWrapper provides the ability find one instance of a remote service and establish a connection to it. It does this by searching Traders looking for all Offers of a particular

service type. Once a connection is established, the connection is reused for subsequent calls. If the connection fails this class begins its search again until it finds a working connection.

5.23.1.1.9 PrimaryFirstOfferWrapper (Class)

This class inherits from OfferWrapper and gives the caller the ability to suggest a preferred service instance whenever service offers are being searched. If that instance is unavailable, the rest of the offers are searched per normal.

5.23.1.1.10 UserManagerWrapper (Class)

The UserManagerWrapper is a singleton class that provides access to a single instance of a remote service type (in this case UserManager) where many instances may exist in the Traders. If the connection to the current instance is lost, it re-establishes the connection, possible with a different instance of the desired service type. This class supports storing properties with values that are string data or binary data.

5.23.1.1.11 WrappedOffer (Class)

A WrappedOffer represents a possible instance of a remote object. The OfferWrapper class holds an array of WrappedOffers and walks the array looking for a valid proxy to use. If the valid proxy subsequently become unavailable, OfferWrapper searches its list of WrappedOffers until it finds another valid WrappedOffer. Failing that, OfferWrapper again searches its Trader list so it can re-populate its WrappedOffer list.

5.24.1.1.1 BasicRequestHandler (Class)

This abstract base class provides an implementation of the `WSRequestHandler.processRequest()` method that provides optional XML validation against specified XSD files and optional digital signature verification as well. It is intended to be used by request handlers that plan to take XML in and return XML to the calling client.

5.24.1.1.2 DataModel (Class)

The data model class serves as a collection of objects. It provides an efficient lookup mechanism for locating any object, and methods which allow for the retrieval of all objects of a particular type. Additionally, this class provides the ability to attach observer objects which are notified when objects are added to or removed from the model. Objects may also notify the `DataModel` that they have been modified. The model will periodically notify all attached observers of the changes to objects in the model.

5.24.1.1.3 DiscoverTrafficEventClassesCmd (Class)

The `DiscoverTrafficEventClassesCmd` class is responsible for discovering `TrafficEvent` and `TrafficEventFactory` corba objects, wrapping those objects in proxy classes and adding those classes to the `DiscoveryManager`'s Object Cache. This class also listens to appropriate corba event channels and updates the Object cache accordingly.

5.24.1.1.4 DiscoveryHost (Class)

This interface defines the methods that the `DiscoveryManager` relies on. It must be implemented by any class that will create a `DiscoveryManager`.

5.24.1.1.5 DiscoveryManager (Class)

This `SystemContextProvider` interface defines some of the functionality required by a class which provides discovery services for CHART services. It is used by both the CHART GUI and the CHART backend services. A class which implements this interface must provide "get" accessor methods for the system profile properties, the data model, and the main processing queue for a service, for instance. It also provides access to the root deployment path and dynamic image path, which is used only by the CHART GUI. For the CHART GUI, this interface is known to be implemented by the `MainServlet`; for the back end CHART services, this interface is known to be implemented by the `DiscoveryManager`.

5.24.1.1.6 ExternalSystemConnectionImpl (Class)

This class knows how to maintain the status of external connections and push them up to the GUI. Also, `ExternalSystemConnectionAlerts` and Notifications can be sent as configured by the admin.

5.24.1.1.7 java.lang.Runnable (Interface)

This interface allows the run method to be called from another thread using Java's threading mechanism.

5.24.1.1.8 ObjectCache (Class)

The ObjectCache is a wrapper for the DataModel. It provides access to DataModel methods to find objects in the data model, delegating those methods to the DataModel itself. It also provides additional methods of finding name filtered objects and discovering "duplicate" objects (as defined by an isDuplicateOf() method of the Duplicatable interface).

5.24.1.1.9 ProxyObject (Class)

This class is a base class for many types of proxy objects store in the CHART object cache (which have been discovered by the DiscoveryDriver tasks), used to provide a standard set of access methods for the proxy objects.

5.24.1.1.10 ProxyTrafficEvent (Class)

The ProxyTrafficEvent object is a proxy for a TrafficEvent corba object which is used to by the DiscoveryManager / ObjectCache. The objects are used to maintain an up to date cache of TrafficEvent data in the object cache for application use.

5.24.1.1.11 QueueableCommand (Interface)

A QueueableCommand is an interface used to represent a command that can be placed on a CommandQueue for asynchronous execution. Derived classes implement the execute method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a CommandQueue. The CommandQueue driver calls the execute method to execute a command in the queue and a call to the interrupted method is made when a CommandQueue is shut down.

5.24.1.1.12 TrafficEventExportHandler (Class)

The TrafficEventExportHandler class is responsible for maintaining up to date Chart TrafficEvent information in the ObjectCache. This data is used to support the class methods which provide data in response to web service requests for exporting Traffic Event data.

5.24.1.1.13 TrafficEventExportModuleProperties (Class)

The TrafficEventExportModuleProperties class provides access methods for properties used by the WSTrafficEventExportModule. It Extends the WebServiceModuleProperties class which allows access to other properties available from the WebService Framework.

5.24.1.1.14TrafficEventExpView (Class)

The TrafficEventExpView class wraps a ProxyTraffic object and provides a view of the proxy object specific to Traffic Event requests. These objects are used by the Velocity Engine which is made available by the WebService Framework. Velocity will apply a defined Traffic Event velocity template to a collection of these objects to generate the XML response to TrafficEvent export requests.

5.24.1.1.15TrafficEventRequestHandler (Class)

The TrafficEventRequestHandler extends the BasicRequestHandler and defines process required to handle TrafficEvent export requests made available by the Chart Export Web Service.

5.24.1.1.16WebService (Class)

This class is the core of each Web Service. It extends the VelocityServlet base class and implements the Service CORBA interface so that Web Service servlets can be administered in the same manner as other CHART service applications.

5.24.1.1.17WebServiceModule (Class)

This interface defines the methods that each module must implement in order to run within the web service framework.

5.24.1.1.18WebServiceModuleProperties (Class)

This abstract base class provides a base for WebServiceModule implementation classes to extend in order to get access to their configuration properties.

5.24.1.1.19WebServiceProperties (Class)

This class provides convenient access to the java Properties object that contains configuration data for the web service and its modules.

5.24.1.1.20WSRequestHandlerSupporter (Class)

This interface defines the methods that will be available to every WSRequestHandler when it is invoked by the framework. It defines the services that the framework will make available to the handlers.

5.24.1.1.21WSTrafficEventExportModule (Class)

The WSTrafficEventExportModule implements the WebServiceModule interface and provides TrafficEvent export functionality via the WebService framework.

5.24.2 SequenceDiagrams

5.24.2.1 TrafficEventExportHandler:getTrafficEventList (Sequence Diagram)

This diagram depicts the processing needed to retrieve TrafficEvent data in response to a specific TrafficEvent export request. If the handler's m_initialized flag is not set throw GeneralException. This in turn will trigger the WebService framework to call the handler's handleProcessingException() method. ProxyTrafficEvent objects are retrieved from the ObjectCache. Based on the optional update window parameter, the functional rights of the client specific token passed in and the owning organization of each proxy object, a collection of appropriate export data is created and returned to the caller. Note: the TrafficEventExpView objects returned are to be used in conjunction with a defined TrafficEventVelocity Template.

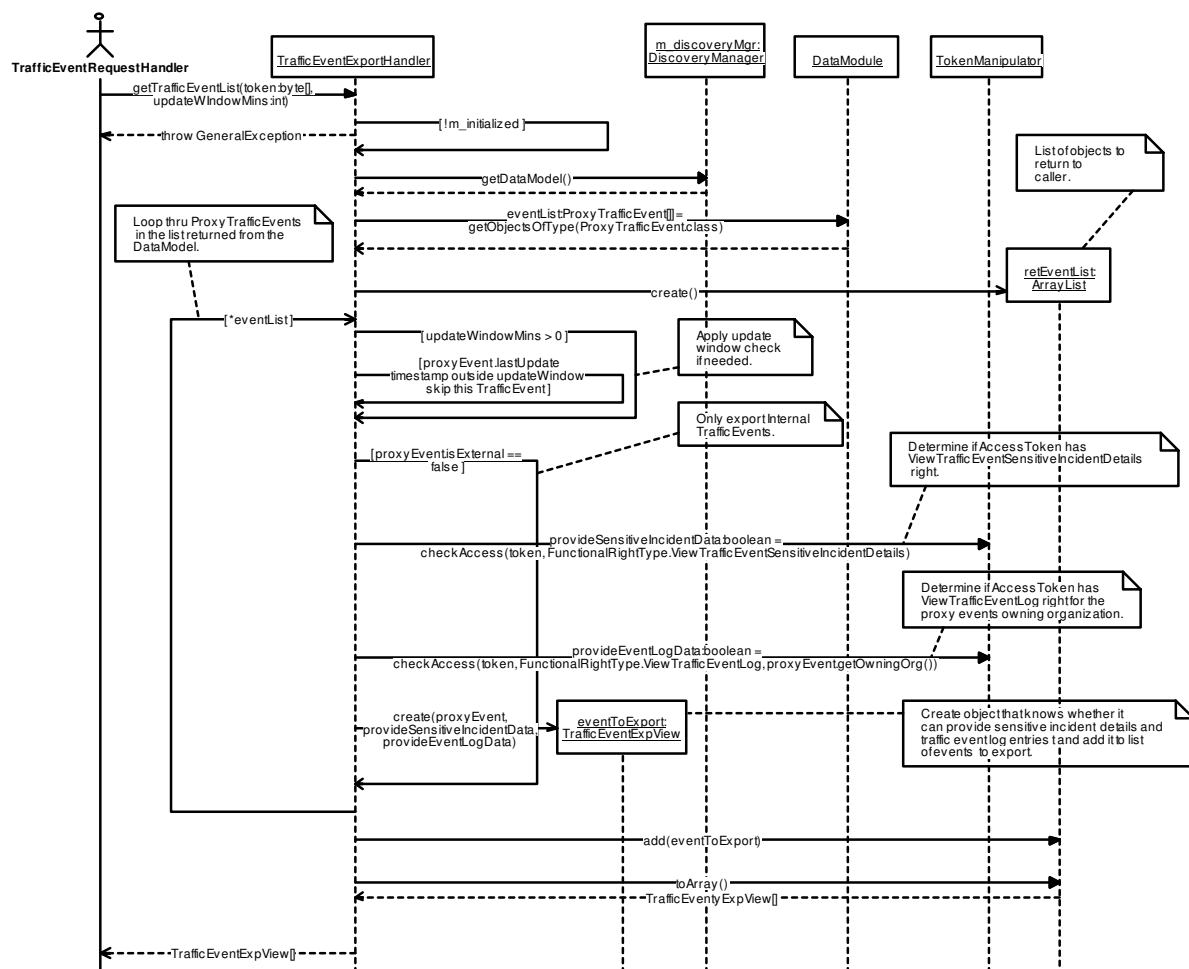


Figure 5-270. TrafficEventExportHandler:getTrafficEventList (Sequence Diagram)

5.24.2.2 TrafficEventHandler:initialize (Sequence Diagram)

This diagram depicts the initialization of the TrafficEventExportHandler class. A DiscoveryTrafficEventClassesCmd is created and added to the DiscoveryManager to populate and maintain TrafficEvent data in the ObjectCache. As a best attempt to make sure the ObjectCache is populated before responding to traffic event web service export requests a thread is started to make sure TrafficEventFactories exist in the ObjectCache before setting initialization flag to true.

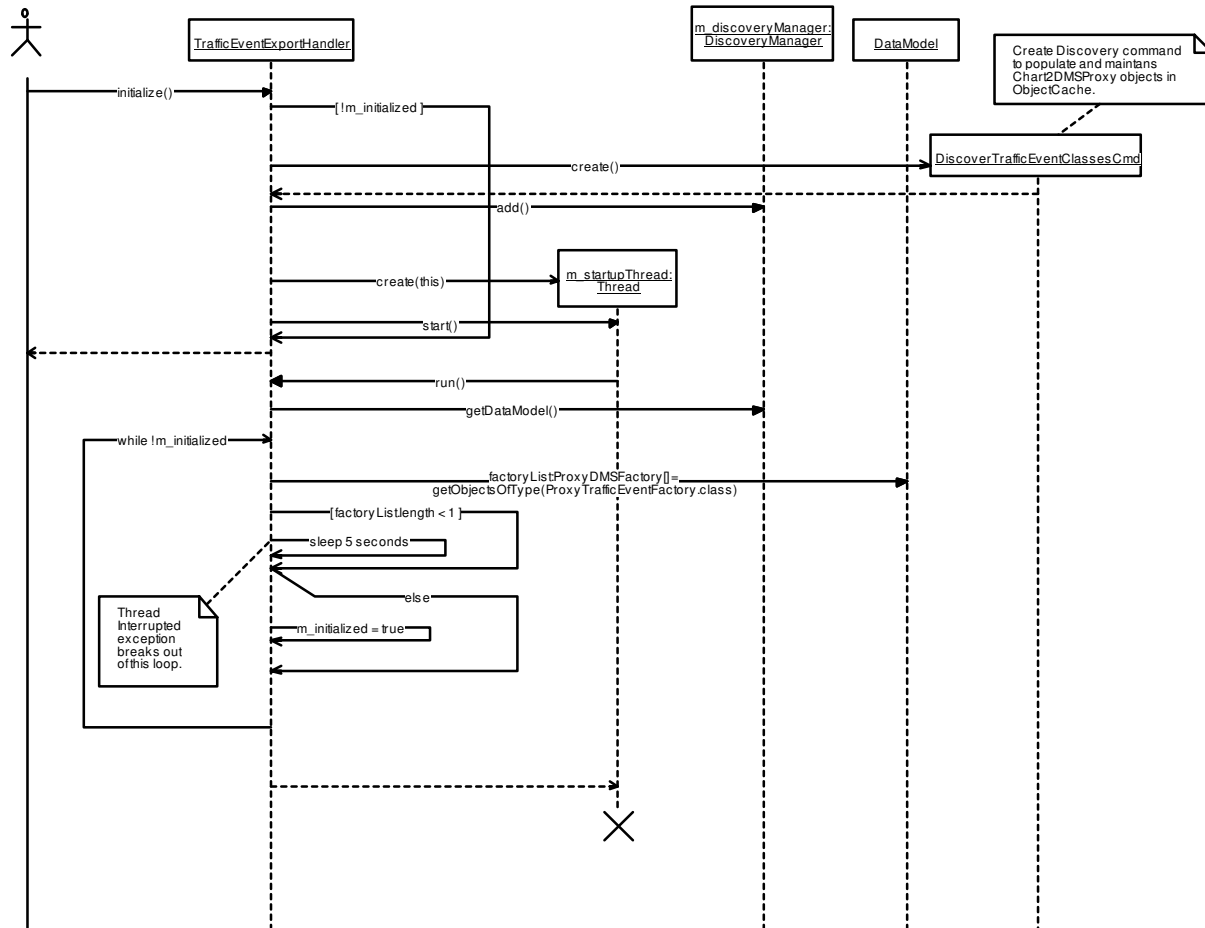


Figure 5-271. TrafficEventHandler:initialize (Sequence Diagram)

5.24.2.3 TrafficEventRequestHandler:processRequest (Sequence Diagram)

This diagram depicts the processing of a Traffic Event Export Request. The processRequest() method of the TrafficEventRequestHandler is called by the RequestManger when a traffic event export request is received by the WebService. The request is handled by getting the appropriate data to export from the TrafficEventExportHandler, adding that data to the Velocity Context passed in and finally returning the path to the correct Velocity Template for the request.

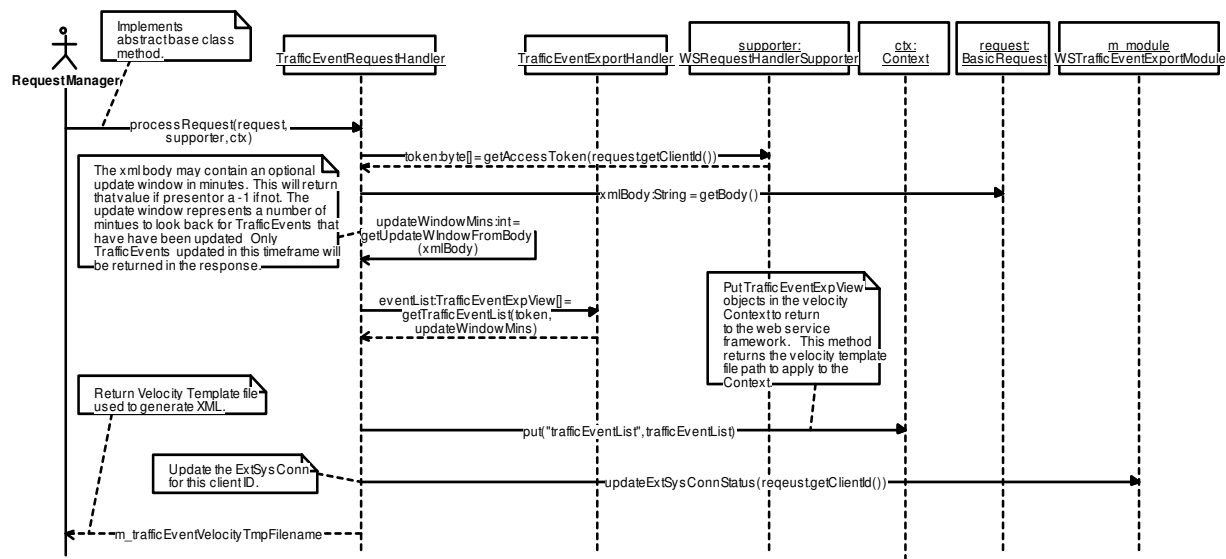


Figure 5-272. TrafficEventRequestHandler:processRequest (Sequence Diagram)

5.24.2.4 WSTrafficEventExportModule:initialize (Sequence Diagram)

This diagram depicts the initialization of the WSTrafficEventExportModule class. A module properties class is created followed by the creation / initialization of the TrafficEventExportHandler class. This class is responsible for maintaining TrafficEvent data in the ObjectCache and providing methods to retrieve traffic event export data in response to export web service requests. Next the TrafficEventRequestHandler is created and registered with the WebService framework to allow the web service URL to start responding to requests.

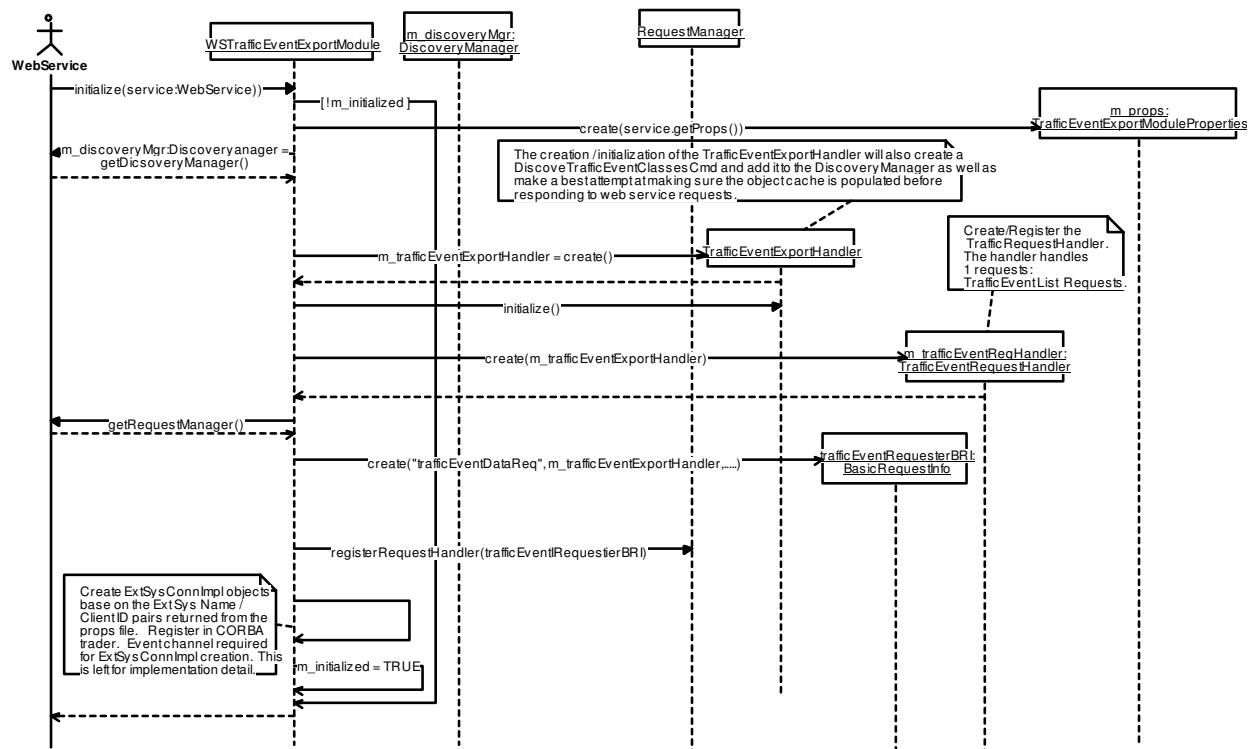


Figure 5-273. WSTrafficEventExportModule:initialize (Sequence Diagram)

5.24.2.5 WSTrafficEventExportModule:shutdown (Sequence Diagram)

The diagram depicts shutdown processing for the WSTrafficEventExportModule. Currently shutdown of the module initiates the TrafficEventExportHandler.shutdown() which cleans up the startupThread if still running. Other cleanup may be determined during implementation.

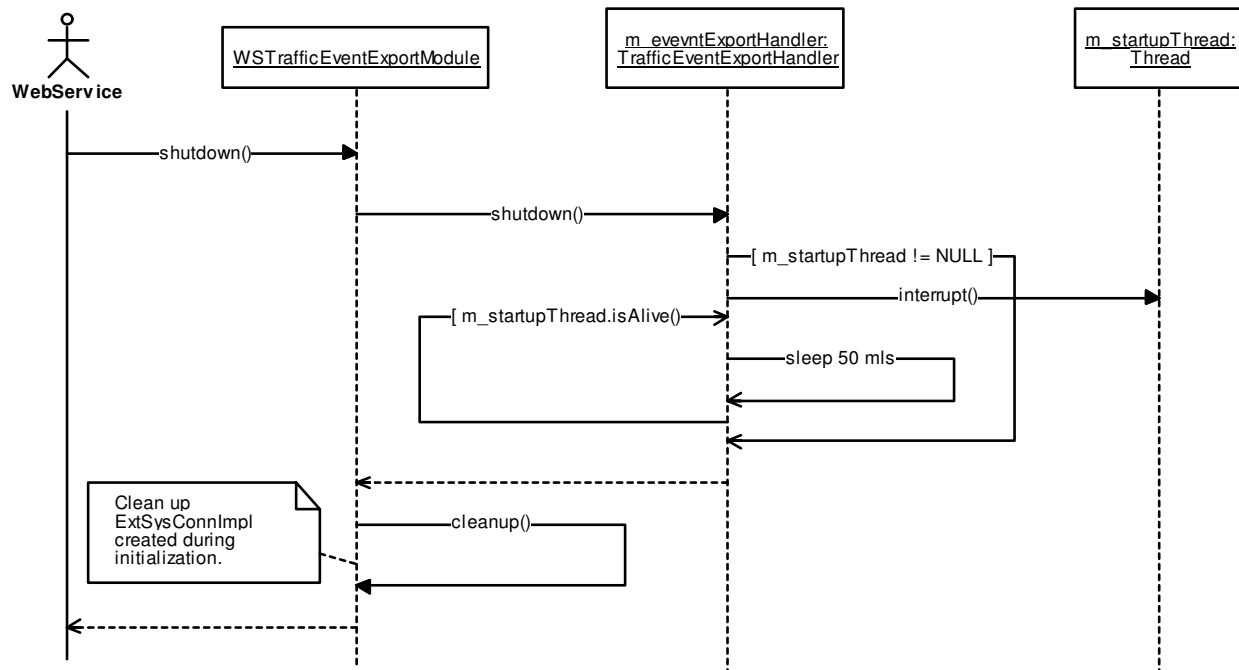


Figure 5-274. WSTrafficEventExportModule:shutdown (Sequence Diagram)

5.25 WSDMSExportModulePkg

5.25.1 Classes

5.25.1.1 WSDMSExportModuleClasses (Class Diagram)

This class diagram defines a `WebServiceModule` used for providing a web service interface for Exporting DMS data. It utilized the Chart `WebService` framework. The `DMSExportHandler` is the main class responsible for maintaining a cache of DMS related objects and providing methods to retrieve information in an exportable form. Note: the `DMSExportHandler` is not `WebService` specific and could be used in the context of the Chart `ServiceApplication` framework if needed.

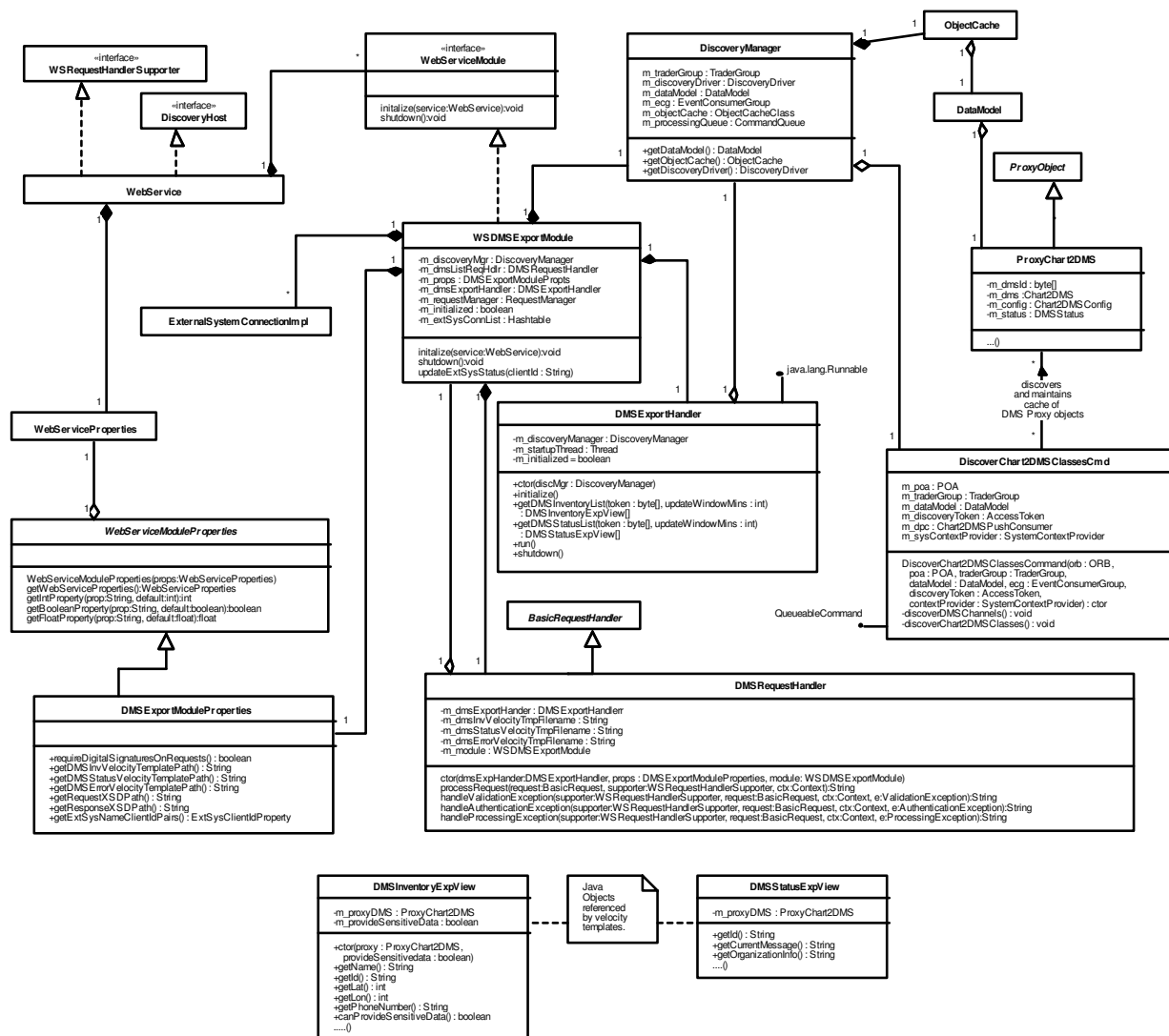


Figure 5-275. WSDMSExportModuleClasses (Class Diagram)

5.25.1.1.1 BasicRequestHandler (Class)

This abstract base class provides an implementation of the `WSRequestHandler.processRequest()` method that provides optional XML validation against specified XSD files and optional digital signature verification as well. It is intended to be used by request handlers that plan to take XML in and return XML to the calling client.

5.25.1.1.2 DataModel (Class)

The data model class serves as a collection of objects. It provides an efficient lookup mechanism for locating any object, and methods which allow for the retrieval of all objects of a particular type. Additionally, this class provides the ability to attach observer objects which are notified when objects are added to or removed from the model. Objects may also notify the `DataModel` that they have been modified. The model will periodically notify all attached observers of the changes to objects in the model.

5.25.1.1.3 DiscoverChart2DMSClassesCmd (Class)

The `DiscoverChart2DMSClassesCmd` class is responsible for discovering `Chart2DMS` and `Chart2DMSFactory` corba objects, wrapping those objects in proxy classes and adding those objects to the `DiscoveryManager`'s Object Cache. This class also listens to appropriate corba events and updates the Object cache accordingly.

5.25.1.1.4 discovers and maintains cache of DMS Proxy objects (Association)

5.25.1.1.5 DiscoveryHost (Class)

This interface defines the methods that the `DiscoveryManager` relies on. It must be implemented by any class that will create a `DiscoveryManager`.

5.25.1.1.6 DiscoveryManager (Class)

This `SystemContextProvider` interface defines some of the functionality required by a class which provides discovery services for CHART services. It is used by both the CHART GUI and the CHART backend services. A class which implements this interface must provide "get" accessor methods for the system profile properties, the data model, and the main processing queue for a service, for instance. It also provides access to the root deployment path and dynamic image path, which is used only by the CHART GUI. For the CHART GUI, this interface is known to be implemented by the `MainServlet`; for the backend CHART services, this interface is known to be implemented by the `DiscoveryManager`.

5.25.1.1.7 DMSExportHandler (Class)

The `DMSExportHandler` class is responsible for maintaining up to date Chart DMS information in the `ObjectCache`. This data is used to support the class methods which provide data in response to web service requests for exporting DMS data.

5.25.1.1.8 DMSExportModuleProperties (Class)

The DMSExportModuleProperties class provides access methods for properties used by the WSDMSExportModule. It Extends the WebServiceModuleProperties class which allows access to other properties available from the WebService Framework.

5.25.1.1.9 DMSInventoryExpView (Class)

The DMSInventoryExpView class wraps a ProxyChart2DMS object and provides a view of the proxy object specific to DMS Inventory requests. These objects are used by the Velocity Engine which is made available by the WebService Framework. Velocity will apply a defined DMS Inventory velocity template to a collection of these objects to generate the XML response to DMS Inventory export requests.

5.25.1.1.10DMSRequestHandler (Class)

The DMSRequestHandler extends the BasicRequestHandler abstract class and implements abstract methods used to handle web service requests for exporting DMS information.

5.25.1.1.11DMSStatusExpView (Class)

The DMSStatusExpView class wraps a ProxyChart2DMS object and provides a view of the proxy object specific to DMS Status requests. These objects are used by the Velocity Engine which is made available by the WebService Framework. Velocity will apply a defined DMS Status velocity template to a collection of these objects to generate the XML response to DMS Status export requests.

5.25.1.1.12ExternalSystemConnectionImpl (Class)

This class knows how to maintain the status of external connections and push them up to the GUI. Also, ExternalSystemConnectionAlerts and Notifications can be sent as configured by the admin.

5.25.1.1.13java.lang.Runnable (Interface)

This interface allows the run method to be called from another thread using Java's threading mechanism.

5.25.1.1.14ObjectCache (Class)

The ObjectCache is a wrapper for the DataModel. It provides access to DataModel methods to find objects in the data model, delegating those methods to the DataModel itself. It also provides additional methods of finding name filtered objects and discovering "duplicate" objects (as defined by an isDuplicateOf() method of the Duplicatable interface).

5.25.1.1.15ProxyChart2DMS (Class)

The ProxyChart2DMS object is a proxy for a Chart2DMS corba object which is used to by the DiscoveryManager / ObjectCache. The objects are used to maintain an up to date cache

of Chart2DMS data in the object cache for application use.

5.25.1.1.16ProxyObject (Class)

5.25.1.1.17QueueableCommand (Interface)

A QueueableCommand is an interface used to represent a command that can be placed on a CommandQueue for asynchronous execution. Derived classes implement the execute method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a CommandQueue. The CommandQueue driver calls the execute method to execute a command in the queue and a call to the interrupted method is made when a CommandQueue is shut down.

5.25.1.1.18WebService (Class)

This class is the core of each Web Service. It extends the VelocityServlet base class and implements the Service CORBA interface so that Web Service servlets can be administered in the same manner as other CHART service applications.

5.25.1.1.19WebServiceModule (Class)

This interface defines the methods that each module must implement in order to run within the web service framework.

5.25.1.1.20WebServiceModuleProperties (Class)

This abstract base class provides a base for WebServiceModule implementation classes to extend in order to get access to their configuration properties.

5.25.1.1.21WebServiceProperties (Class)

This class provides convenient access to the java Properties object that contains configuration data for the web service and its modules.

5.25.1.1.22WSDMSExportModule (Class)

The WSDMSExportModule implements the WebServiceModule interface and provides DMS export functionality via the WebService framework.

5.25.1.1.23WSRequestHandlerSupporter (Class)

This interface defines the methods that will be available to every WSRequestHandler when it is invoked by the framework. It defines the services that the framework will make available to the handlers.

5.25.2 Sequence Diagrams

5.25.2.1 DMSExportHandler:getDMSInventoryList (Sequence Diagram)

This diagram depicts the processing needed to retrieve DMS Inventory data in response to a specific DMS Inventory export request. If the handler's `m_initialized` flag is not set throw `GeneralException`. This in turn will trigger the WebService framework to call the handler's `handleProcessingException()` method. ProxyChart2DMS objects are retrieved from the ObjectCache. Based on the optional update window parameter, the functional rights of the client specific token passed in and the owning organization of each proxy object, a collection of appropriate export data is created and returned to the caller. Note: the `DMSInventoryExpView` objects returned are to be used in conjunction with a defined DMSInventory Velocity Template to generate the XML response to the request. Note: if the handler's `m_initialized` flag is not set throw `GeneralException`. This in turn will trigger the WebService framework to call the handler's `handleProcessingException()` method.

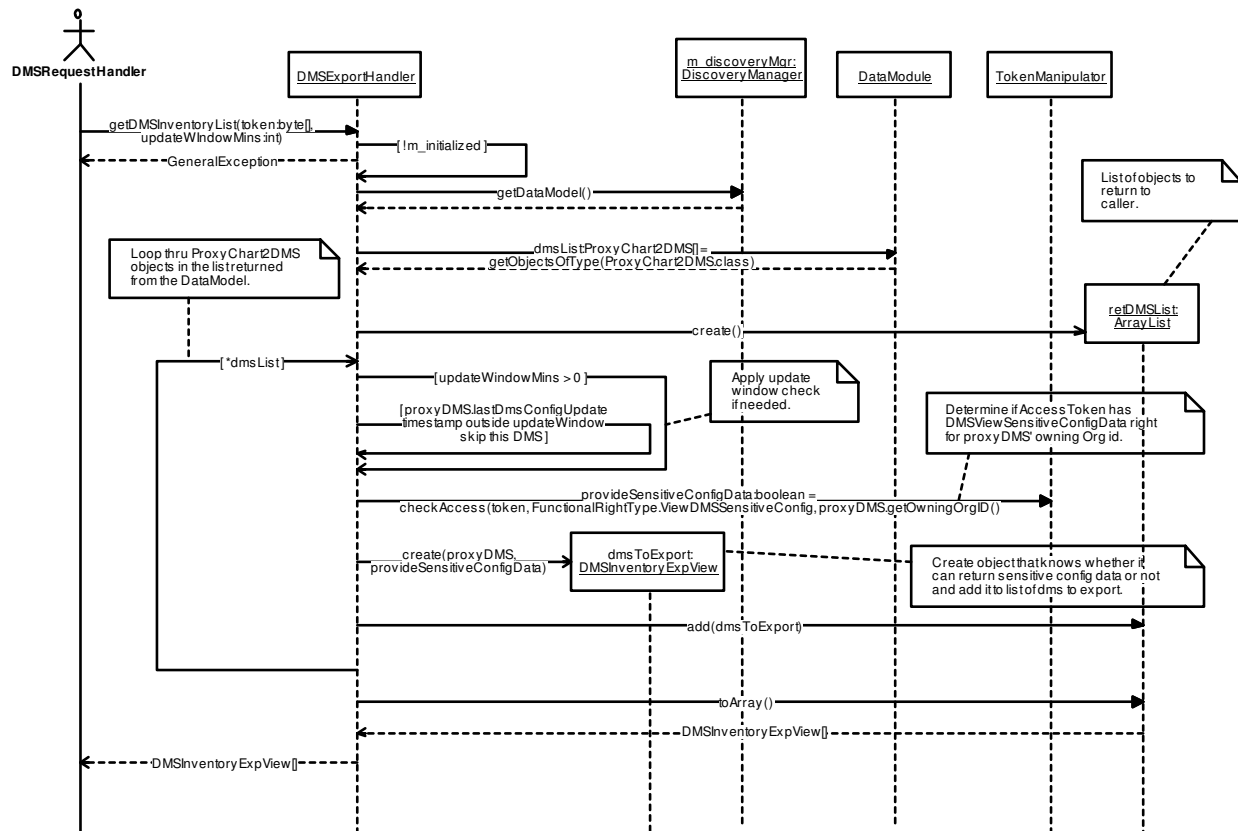


Figure 5-276. DMSExportHandler:getDMSInventoryList (Sequence Diagram)

5.25.2.2 DMSExportHandler:getDMSStatusList (Sequence Diagram)

This diagram depicts the processing needed to retrieve DMS Status data in response to a specific DMS Status export request. If the handler's m_initialized flag is not set throw GeneralException. This in turn will trigger the WebService framework to call the handler's handleProcessingException() method. ProxyChart2DMS objects are retrieved from the ObjectCache. Based on the optional update window parameter a collection of appropriate export data is created and returned to the caller. Note: the DMSStatusExpView objects returned are to be used in conjunction with a defined DMSStatus Velocity Template to generate the XML response to the request.

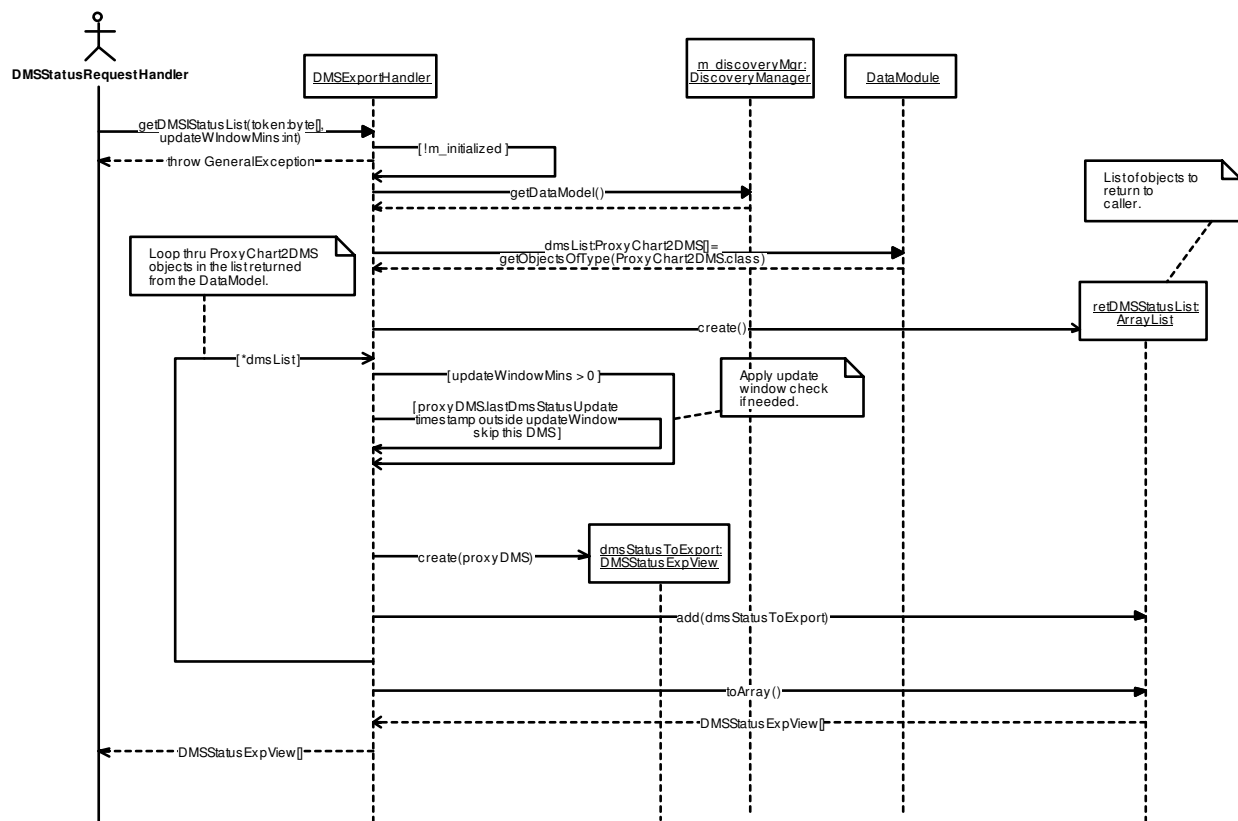


Figure 5-277. DMSExportHandler:getDMSStatusList (Sequence Diagram)

5.25.2.3 DMSExportHandler:initialize (Sequence Diagram)

This diagram depicts the initialization of the DMSExportHandler class. A DiscoveryChart2DMSClassesCmd is created and added to the DiscoveryManager to populate and maintain DMS data in the ObjectCache. As a best attempt to make sure the ObjectCache is populated before responding to dms web service export requests a thread is started to make sure Chart2DMSFactories exist in the ObjectCache before setting initialization flag to true.

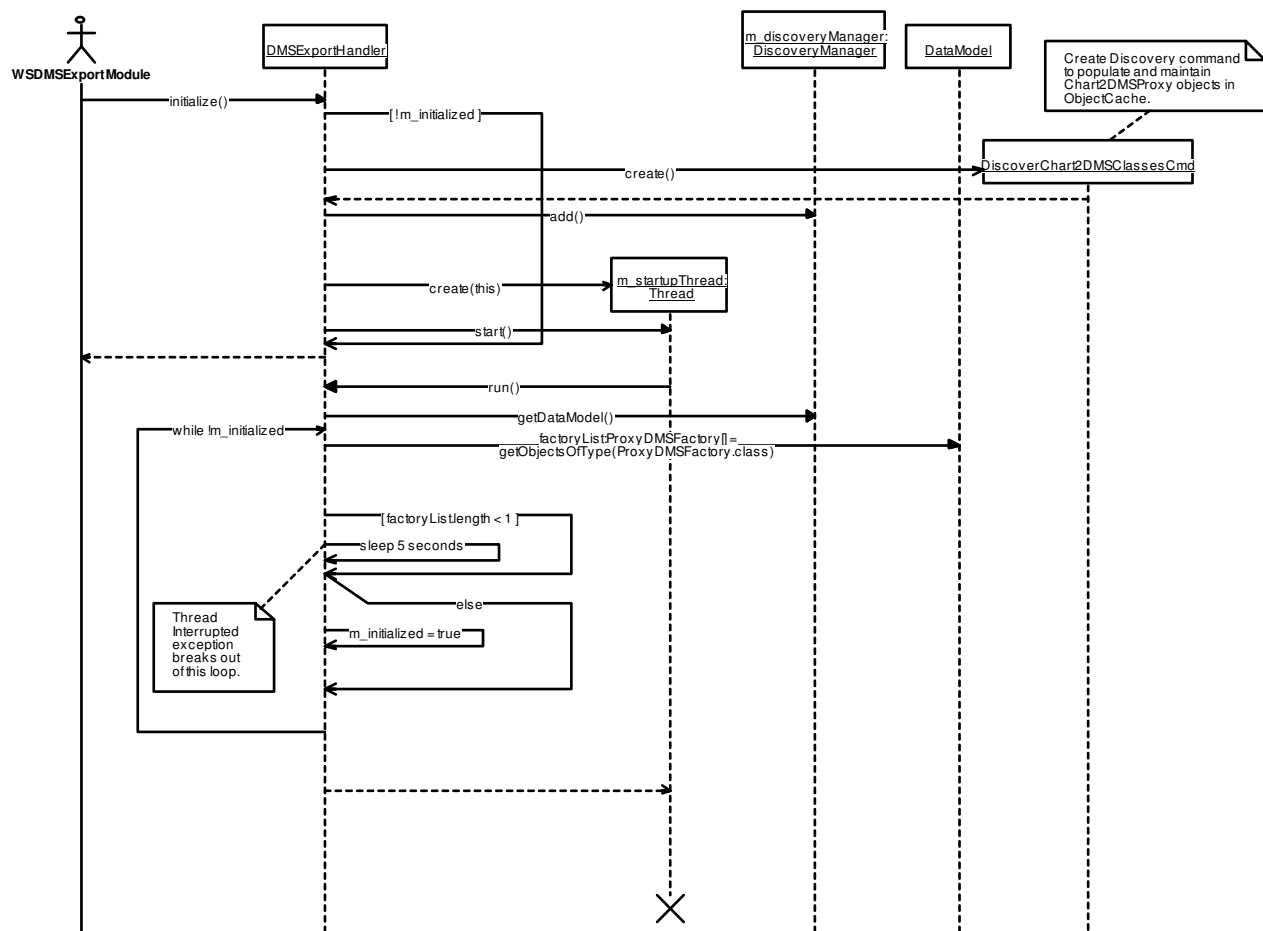


Figure 5-278. DMSExportHandler:initialize (Sequence Diagram)

5.25.2.4 DMSRequestHandler:handleExceptions (Sequence Diagram)

This sequence diagram depicts Exception handling done as part of the WebService framework. The DMSRequestHandler derives from the BasicRequestHandler class and implements 3 abstract methods (handleValidationException(), handleAuthenticationException() and handleProcessingException()). These methods are called from the WebService frame work when exceptions are encountered. Each method retrieves information as need from the arguments passed in and creates a response using a Velocity Context object and a predefine Velocity template. Note: for handleValidationException() if inbound xml validation fails, pass the invalid XML string back in the XML response defined as CDATA so it will not be parsed.

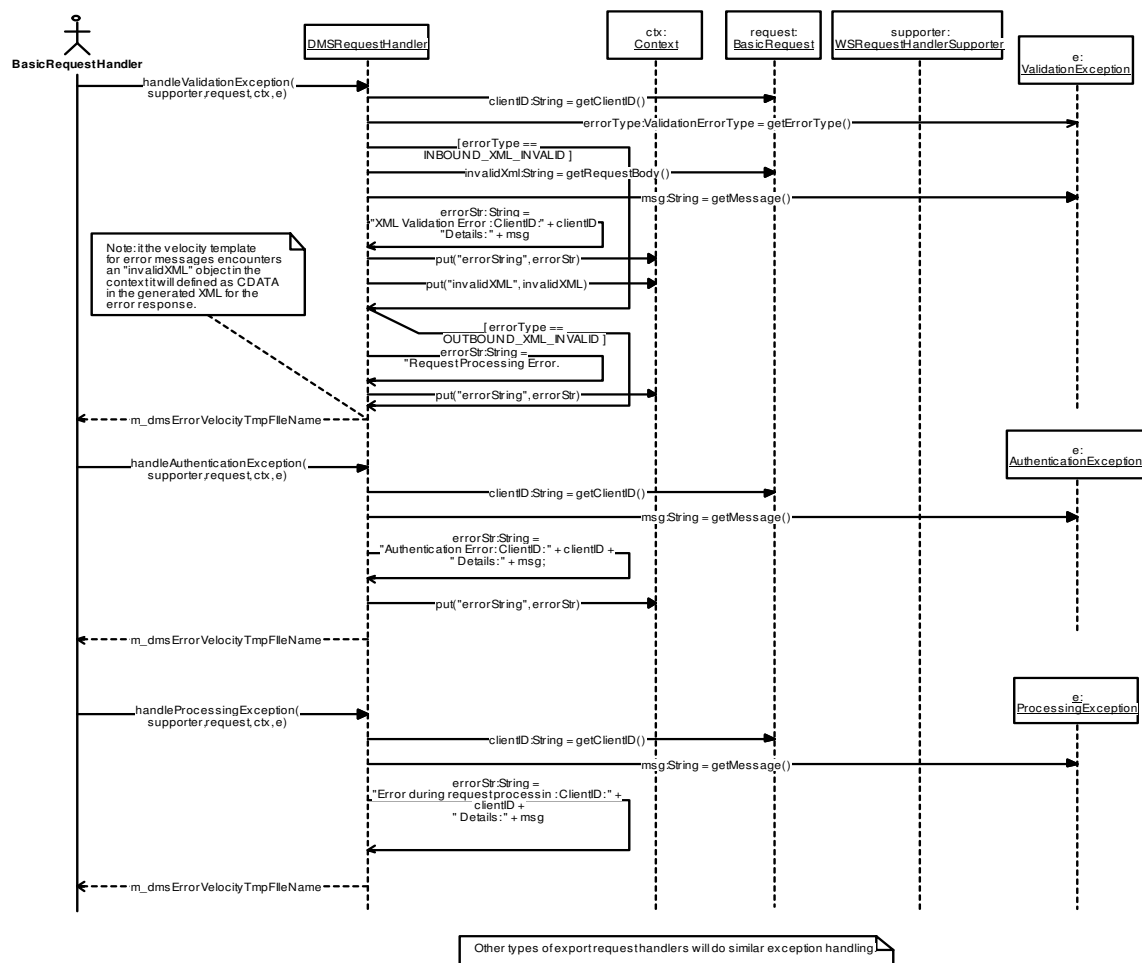


Figure 5-279. DMSRequestHandler:handleExceptions (Sequence Diagram)

5.25.2.5 DMSRequestHandler:processRequest (Sequence Diagram)

This diagram depicts the processing of a DMS Export Request. The processRequest() method of the DMSRequestHandler is called by the RequestManger when a dms export request is received by the WebService. The request type is retrieved from the xml body of the request. Two types of request are handled. DMSInventory and DMSStatus request. The requests are handled by getting the appropriate data to export from the DMSExportHandler, adding that data to the Velocity Context passed in and finally returning the path to the correct Velocity Template for the request.

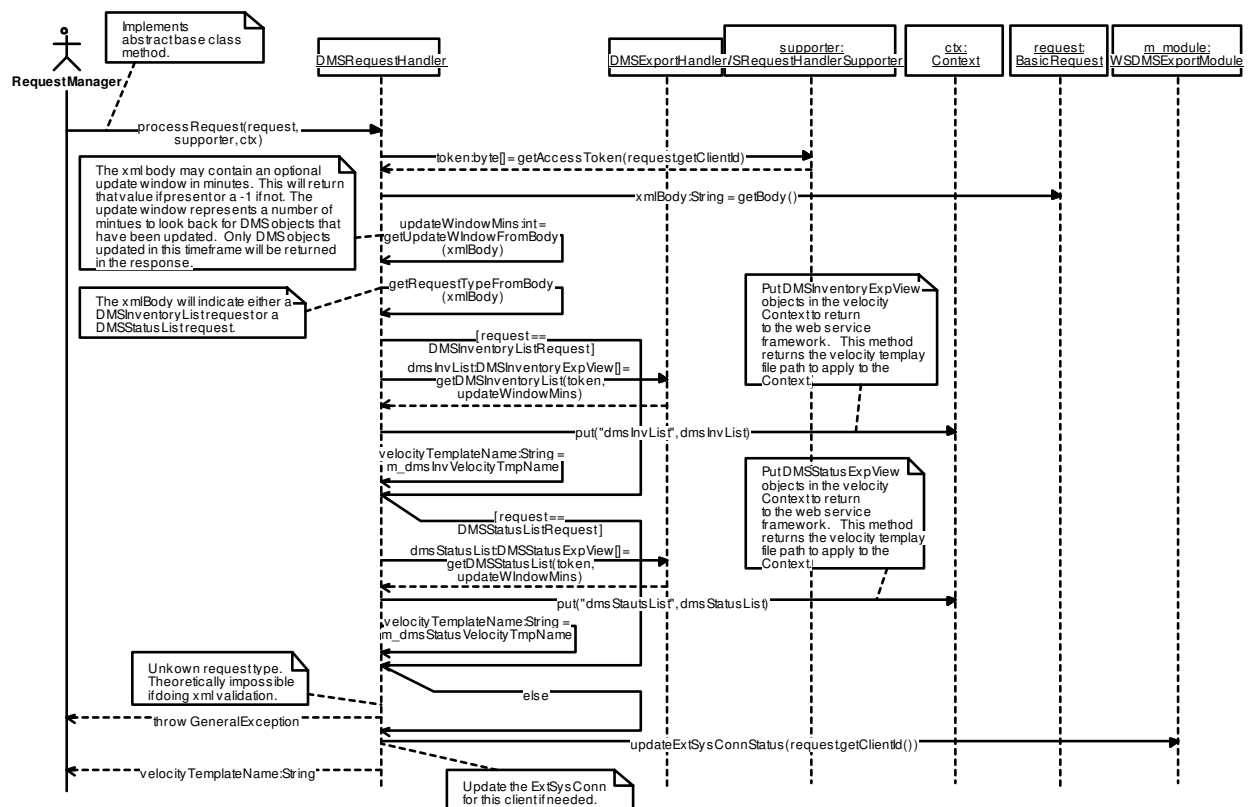


Figure 5-280. DMSRequestHandler:processRequest (Sequence Diagram)

5.25.2.6 WSDMSExportModule:initialize (Sequence Diagram)

This diagram depicts the initialization of the WSDMSExportModule class. A module properties class is created followed by the creation / initialization of the DMSExportHandler class. This class is responsible for maintaining DMS data in the ObjectCache and providing methods to retrieve dms export data in response to dms export web service requests. Next the DMSRequestHandler is created and registered with the WebService framework to allow the web service URL to start responding to requests.

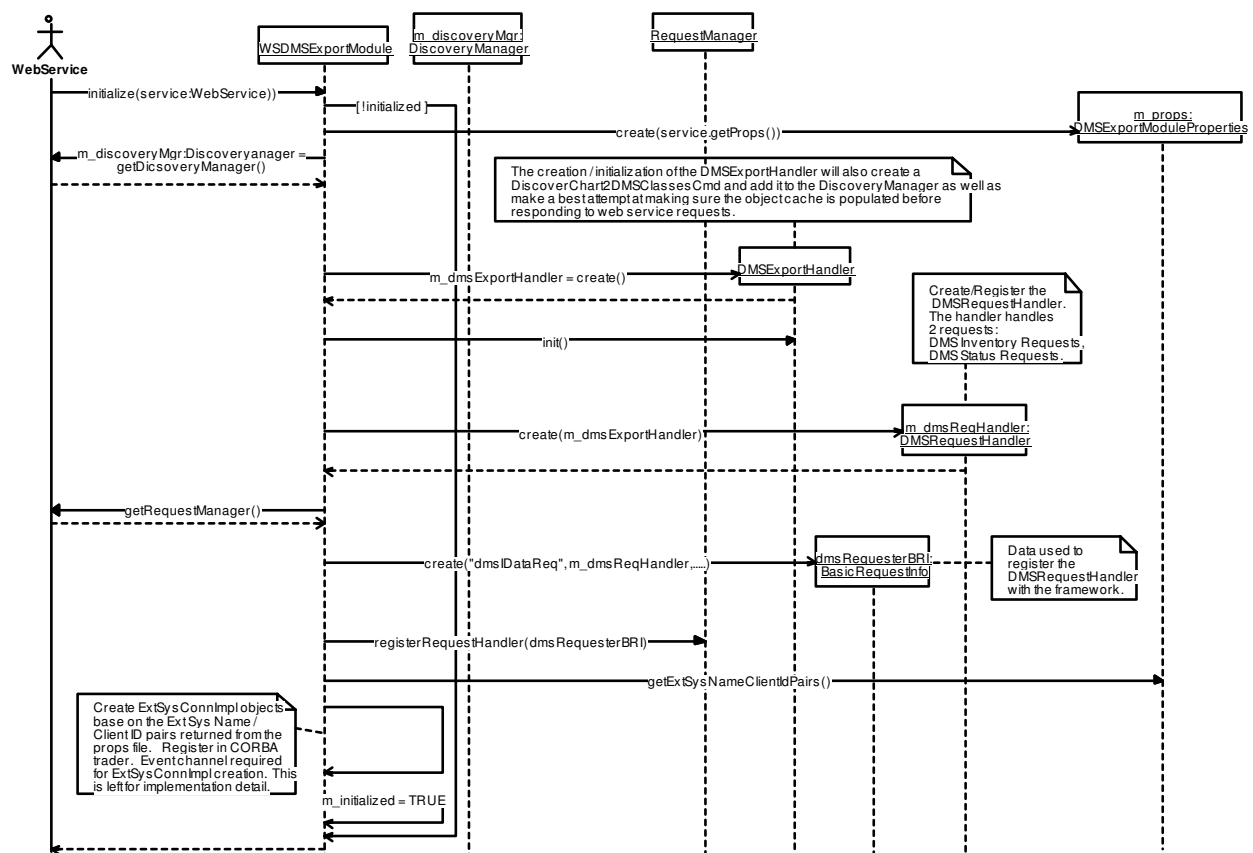


Figure 5-281. WSDMSExportModule:initialize (Sequence Diagram)

5.25.2.7 WSDMSEExportModule:shutdown (Sequence Diagram)

The diagram depicts shutdown processing for the WSDMSEExportModule. Currently shutdown of the module initiates the DMSEExportHandler.shutdown() which cleans up the startupThread if still running. Other cleanup may be determined during implementation.

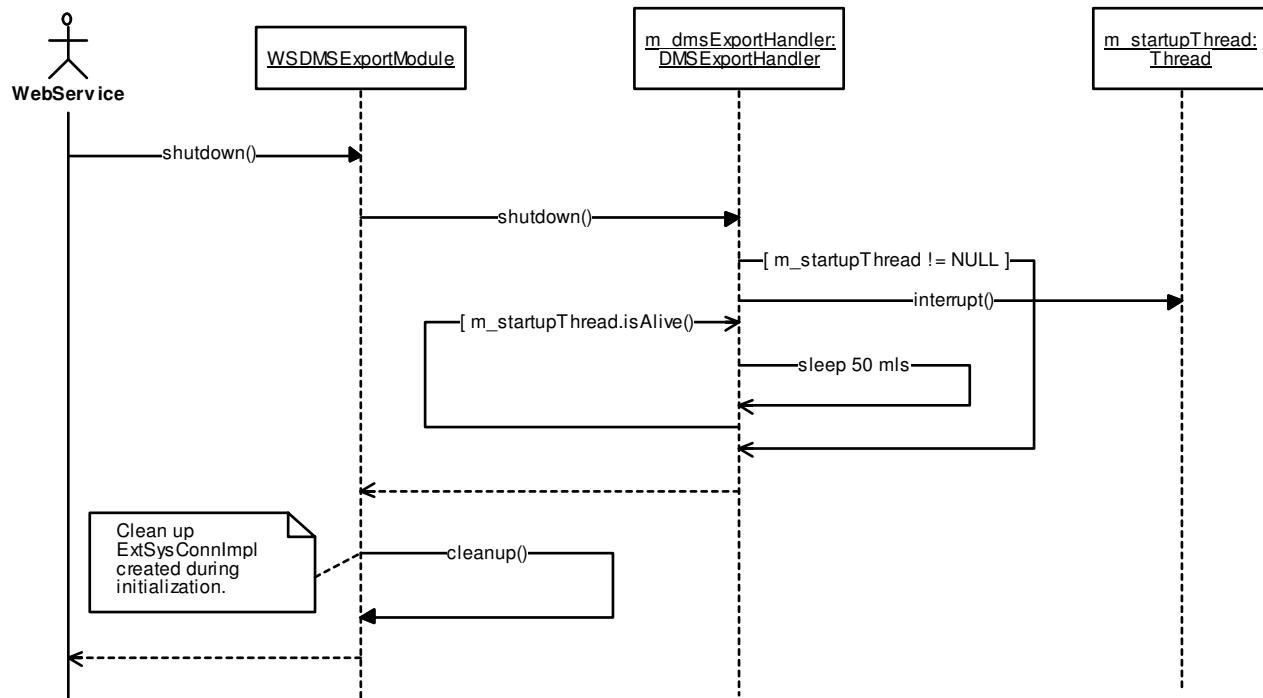


Figure 5-282. WSDMSEExportModule:shutdown (Sequence Diagram)

5.26 WebServices.TollrateImportModule

5.26.1 Classes

5.26.1.1 TollRateImportModuleClasses (Class Diagram)

This diagram shows the classes used by the tollrateimportmodule.

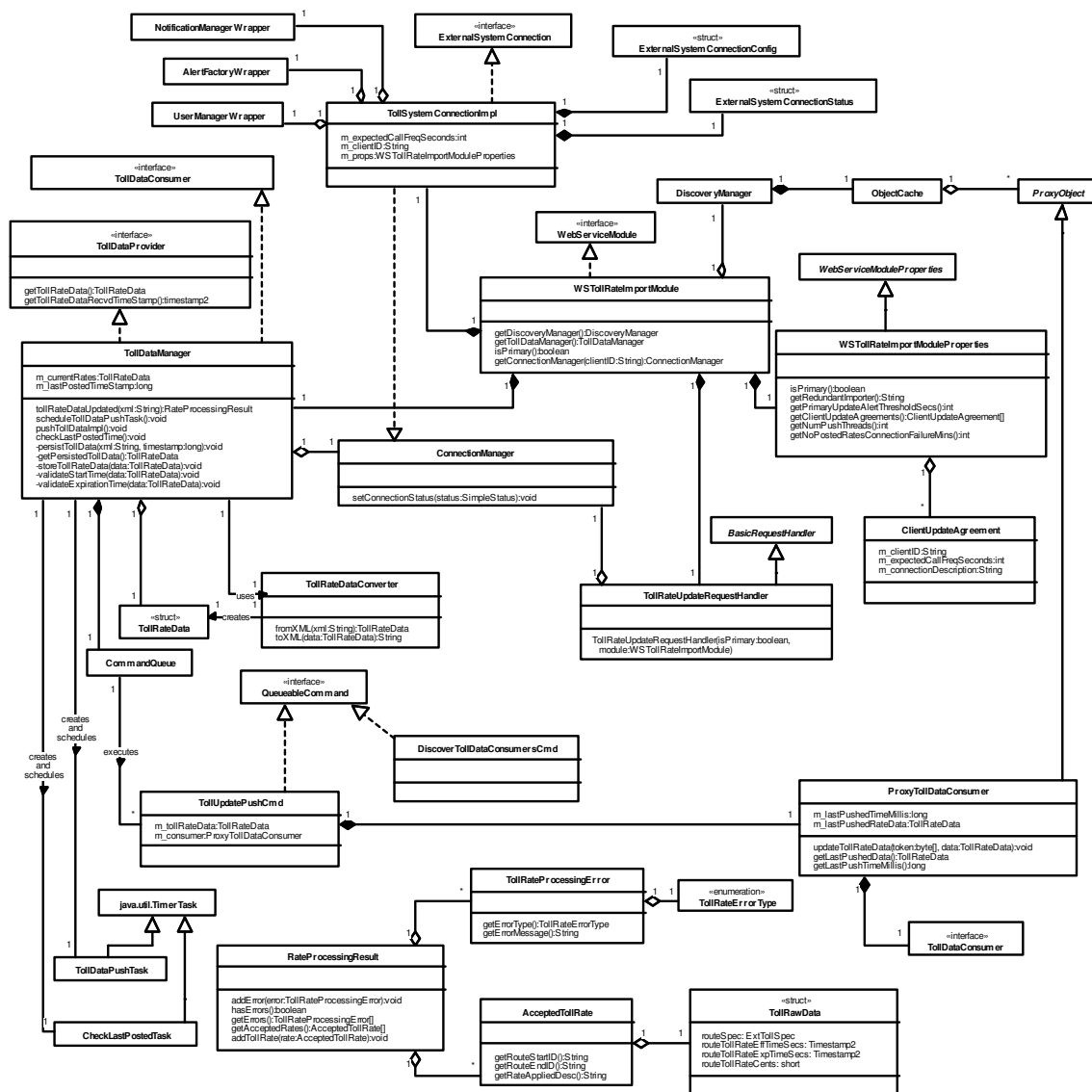


Figure 5-283. TollRateImportModuleClasses (Class Diagram)

5.26.1.1.1 AcceptedTollRate (Class)

This class stores data about a posted toll rate that was accepted by the CHART system. It is used to format the response XML to return to the posting client.

5.26.1.1.2 AlertFactoryWrapper (Class)

This singleton class provides a wrapper for the Alert Factory that provides automatic location of an Alert Factory and automatic re-discovery should the Alert Factory reference return an error. This class also allows for built-in fault tolerance by automatically failing over to a "working" Alert Factory without the user of this class being aware that this being done. In addition, this class defers the discovery of the Alert Factory until its first use, thus eliminating a start-up dependency for modules that use the Alert Factory.

This class delegates all of its method calls to the system AlertFactory using its currently known good reference to an AlertFactory. If the current reference returns a CORBA failure in the delegated call, this class automatically switches to another reference. When there are no good references (as is true the first time the object is used), this class issues a trader query to (re)discover the published Alert Factory objects in the system. During a method call, the trader will be queried at most one time and under normal circumstances, not at all.

5.26.1.1.3 BasicRequestHandler (Class)

This abstract base class provides an implementation of the WSRequestHandler.processRequest() method that provides optional XML validation against specified XSD files and optional digital signature verification as well. It is intended to be used by request handlers that plan to take XML in and return XML to the calling client.

5.26.1.1.4 CheckLastPostedTask (Class)

This class implements the java.util.TimerTask interface, and is scheduled on a java.util.Timer object to check the last time a toll rate update was received from the toll rate supplier. If there is too long of a delay, a FAILURE condition is posted to the external connection, and an alert and/or notification is/are posted according to the configuration of the toll rate external connection.

5.26.1.1.5 ClientUpdateAgreement (Class)

This class stores information about a client connection agreement for this service. The service will setup an external system connection for each ClientUpdateAgreement configured in the properties file.

5.26.1.1.6 CommandQueue (Class)

The CommandQueue class provides a queue for QueueableCommand objects. The CommandQueue has a thread that it uses to process each QueueableCommand in a first in first out order. As each command object is pulled off the queue by the CommandQueue's thread, the command object's execute method is called, at which time the command

performs its intended task.

5.26.1.1.7 ConnectionManager (Class)

This interface must be implemented by any class that manages external system connections.

5.26.1.1.8 DiscoverTollDataConsumersCmd (Class)

This class is responsible for discovering TollDataConsumer objects and storing proxies for them in the cache.

5.26.1.1.9 DiscoveryManager (Class)

This SystemContextProvider interface defines some of the functionality required by a class which provides discovery services for CHART services. It is used by both the CHART GUI and the CHART backend services. A class which implements this interface must provide "get" accessor methods for the system profile properties, the data model, and the main processing queue for a service, for instance. It also provides access to the root deployment path and dynamic image path, which is used only by the CHART GUI. For the CHART GUI, this interface is known to be implemented by the MainServlet; for the back end CHART services, this interface is known to be implemented by the Discovery Manager.

5.26.1.1.10 ExternalSystemConnection (Class)

This interface defines external connections and provides status reporting.

5.26.1.1.11 ExternalSystemConnectionConfig (Class)

This struct defines a connection to an external system.

5.26.1.1.12 ExternalSystemConnectionStatus (Class)

This struct is used to report status for an ExternalSystemConnection.

5.26.1.1.13 java.util.TimerTask (Class)

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

5.26.1.1.14 NotificationManagerWrapper (Class)

This singleton class presents the same interface as the NotificationManager, but uses a FirstAvailableOfferWrapper to provide fault tolerant access to the methods.

5.26.1.1.15 ObjectCache (Class)

The ObjectCache is a wrapper for the DataModel. It provides access to DataModel methods to find objects in the data model, delegating those methods to the DataModel itself. It also provides additional methods of finding name filtered objects and discovering

"duplicate" objects (as defined by an `isDuplicateOf()` method of the `Duplicatable` interface).

5.26.1.1.16ProxyObject (Class)

This class is a base class for many types of proxy objects store in the CHART object cache (which have been discovered by the `DiscoveryDriver` tasks), used to provide a standard set of access methods for the proxy objects.

5.26.1.1.17ProxyTollDataConsumer (Class)

This class serves as a proxy for a `TollDataConsumer`. It keeps track of the last time we pushed data to a consumer and what data was pushed.

5.26.1.1.18QueueableCommand (Class)

A `QueueableCommand` is an interface used to represent a command that can be placed on a `CommandQueue` for asynchronous execution. Derived classes implement the `execute` method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a `CommandQueue`. The `CommandQueue` driver calls the `execute` method to execute a command in the queue and a call to the `interrupted` method is made when a `CommandQueue` is shut down.

5.26.1.1.19RateProcessingResult (Class)

This class stores the results of processing a posted toll rate update.

5.26.1.1.20TollDataConsumer (Class)

CORBA interface that must be implemented by any CHART component that wants to be notified when toll rates changes.

5.26.1.1.21TollDataManager (Class)

This class manages the current toll rate data for this service. It keeps track of the last known toll rates and the time they were received.

5.26.1.1.22TollDataProvider (Class)

System interface that is implemented by services that provide toll rate data to the system.

5.26.1.1.23TollDataPushTask (Class)

This class implements the `java.util.TimerTask` interface, and is scheduled on a `java.util.Timer` object to move pushing of toll rate data to `TollDataConsumer` objects onto a background thread to minimized the effect of CORBA communication delays on this import module.

5.26.1.1.24TollRateData (Class)

This struct provides a collection of TollRawData objects along with a timestamp of when they were posted by the toll rate source.

5.26.1.1.25TollRateDataConverter (Class)

This class is responsible for converting posted XML toll rate documents to CORBA structures used internally by CHART components and converting the internal structures back to XML.

5.26.1.1.26TollRateErrorType (Class)

Enumeration of toll rate error types.

5.26.1.1.27TollRateProcessingError (Class)

This class stores information about an error encountered while processing a toll rate update.

5.26.1.1.28TollRateUpdateRequestHandler (Class)

This class is the request handler that is responsible for handling posted XML toll rate documents.

5.26.1.1.29TollRawData (Class)

This structure is used by the CHART importer of toll rates to pass toll rate data into the travel route factory. The intent is that the importer will receive all toll rates in one transaction and will pass all toll rates on to the TravelRouteFactory in one transaction. In R3B3, Vector was added as a toll rate provider for the CHART system.

5.26.1.1.30TollSystemConnectionImpl (Class)

This class represents an external system connection that the toll rate import module expects toll rates to be posted from. It stores the id of the client application that is expected to post data and how frequently they are expected to post.

5.26.1.1.31TollUpdatePushCmd (Class)

This class performs the asynchronous push of toll rate data to a consumer.

5.26.1.1.32UserManagerWrapper (Class)

The UserManagerWrapper is a singleton class that provides access to a single instance of a remote service type (in this case UserManager) where many instances may exist in the Traders. If the connection to the current instance is lost, it re-establishes the connection, possible with a different instance of the desired service type. This class supports storing properties with values that are string data or binary data.

5.26.1.1.33WebServiceModule (Class)

This interface defines the methods that each module must implement in order to run within the web service framework.

5.26.1.1.34WebServiceModuleProperties (Class)

This abstract base class provides a base for WebServiceModule implementation classes to extend in order to get access to their configuration properties.

5.26.1.1.35WSTollRateImportModule (Class)

This class is the pluggable web service module that provides toll rate import functionality.

5.26.1.1.36WSTollRateImportModuleProperties (Class)

This class provides convenience methods for accessing configuration file properties specific to the toll rate import module.

5.26.2 Sequence Diagrams

5.26.2.1 CHART2.webservices.tollrateimportmodule:TollDataManager.tollRateDataUpdated (Sequence Diagram)

This diagram shows the processing that is performed when the TollDataManager.tollRateDataUpdated method is invoked to pass new toll rate XML that has been received. First a RateProcessingResult is created to store the results of the operation for return. Next the XML is converted using the TollRateDataConverter. If there is an error converting the XML it is returned in the RateProcessingResult. Otherwise the start time in the XML is validated. If the start time is invalid (more than a configurable number of minutes in the future) an error is returned. Otherwise the expiration time in the XML is validated. If the expiration time is invalid (not later than the start date/time) an error is returned. If everything is valid the toll rate data is stored in memory as our latest and the received XML is persisted to a flat file. If everything worked correctly a TollDataPushTask is scheduled for immediate execution to push the new toll rate date to all consumers.

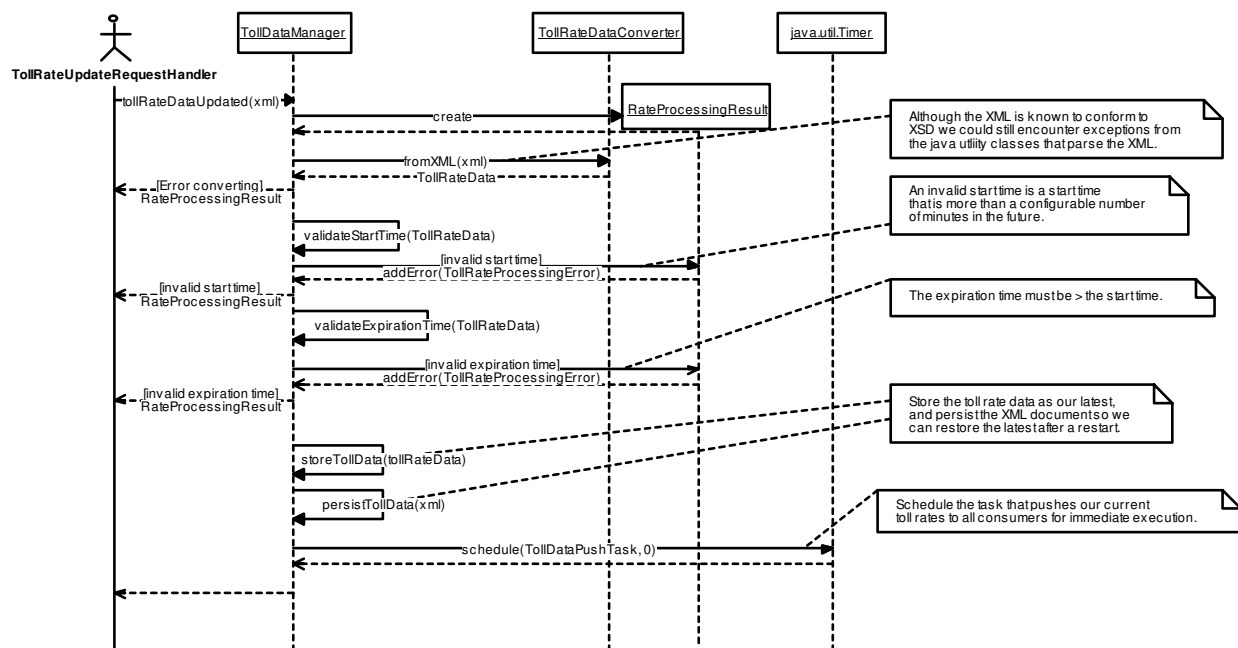


Figure 5-284.

CHART2.webservices.tollrateimportmodule:TollDataManager.tollRateDataUpdated (Sequence Diagram)

5.26.2.2 CHART2.webservices.tollrateimportmodule:TollDataManager.updateTollRateData (Sequence Diagram)

This diagram shows the processing performed by the TollDataManager when the updateTollRateData method (of the TollDataConsumer CORBA interface) is invoked by a remote process. If the current data stored by this TollDataManager was received later than the passed TollRateData then the data we have been sent is not the latest and will be ignored (it is coming from an asynchronous push with retries). If the passed TollRateData does have a more recent timestamp that the currently stored data the passed it is stored as our current, then converted to XML via the TollRateDataConverter and finally the converted XML is persisted to a flat file for process recovery purposes. Note that the data is not pushed to TollDataConsumers because it was not received directly from the source.

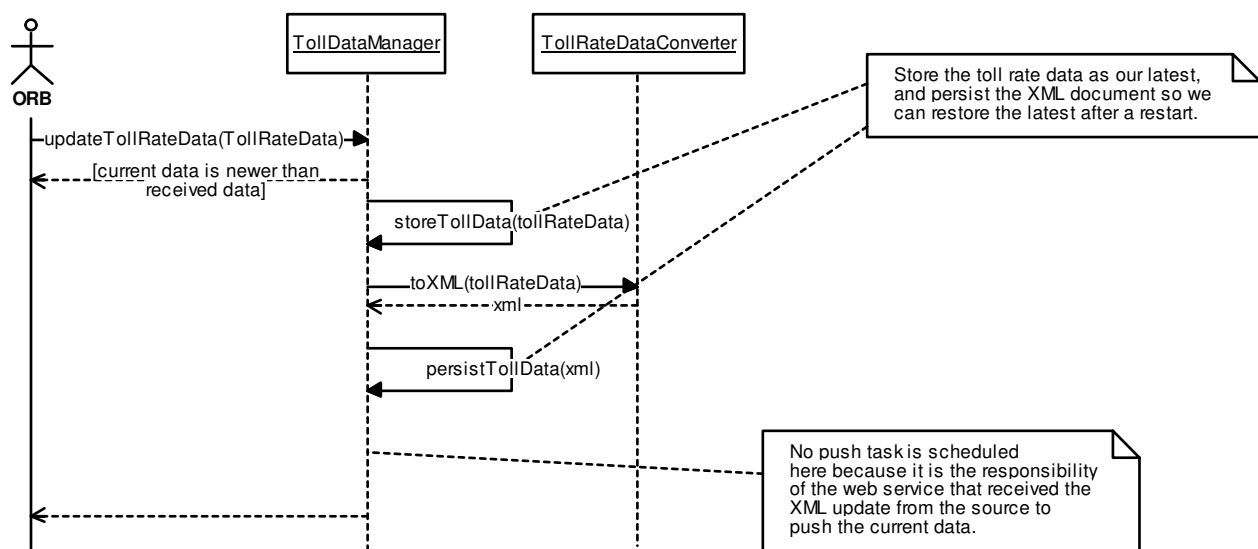


Figure 5-285.

CHART2.webservices.tollrateimportmodule:TollDataManager.updateTollRateData (Sequence Diagram)

5.26.2.3 CHART2.webservices.tollrateimportmodule:TollDataPushTask.run (Sequence Diagram)

This diagram shows the processing performed by the TollDataPushTask in order to push current TollRateData to all TollDataConsumers. When the timer fires, the TollDataPushTask calls the TollDataManager.pushTollDataImpl method which retrieves all ProxyTollDataConsumer objects from the DataModel. The TollDataManager then clears all currently queued TollUpdatePushCmd updates from the CommandQueue to ensure that all previously scheduled asynchronous toll rate pushes are abandoned. It then loops over all ProxyTollDataConsumer objects and checks if the last TollRateData that we pushed to them was received earlier than the data we are currently pushing. If it was, then a push is needed and a TollUpdatePushCmd is created and added to the CommandQueue. At some point in the future (when the CommandQueue has a thread available), the TollUpdatePushCmd.execute() method is invoked by the CommandQueue. The TollUpdatePushCmd performs a second check to ensure that the data we are pushing is more recent than the data we last pushed to this same consumer. If it is, the TollDataConsumer.updateTollRateData method is invoked to pass them the updated data. The consumer returns an array of routes that were expected to have toll rate data in the update but did not. If this array is not empty the ConnectionManager is called to set the external connection status to WARNING level.

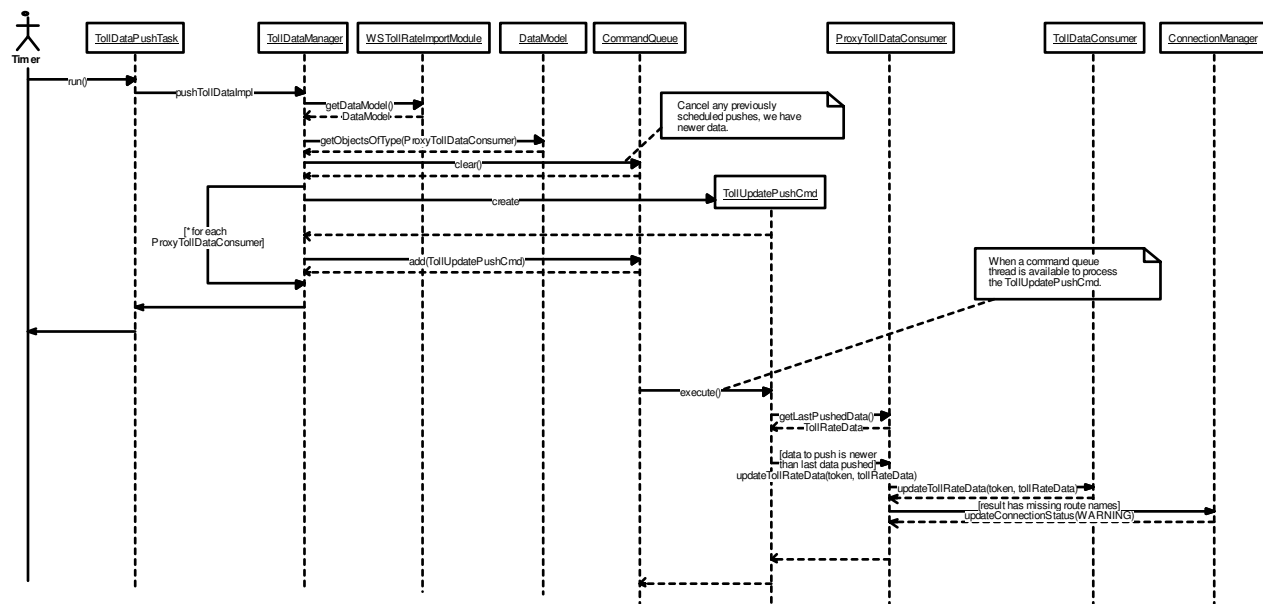


Figure 5-286. CHART2.webservices.tollrateimportmodule:TollDataPushTask.run (Sequence Diagram)

5.26.2.4 CHART2.webservices.tollrateimportmodule:TollRateUpdateRequestHandler.processRequest (Sequence Diagram)

This diagram shows the processing performed when the toll rate import modules receives a web service request to update the current toll rates. The BasicRequestHandler abstract base class is invoked by the WebService core. Before calling the TollRateUpdateRequestHandler the base class will verify the digital signature used to sign the request and will validate the XML against the XSD. If all is valid the base class invokes the processRequest method of the TollRateUpdateRequestHandler. The handler immediately checks if it is the primary or the backup service. If it is not primary the ConnectionManager is obtained and a connection alert is raised. Next the request body XML is obtained and passed to the TollDataManager who returns a RateProcessingResult which is placed in the context and used for the creation of the response XML. If the returned RateProcessingResult contains errors the error response template will be used, otherwise if this service is the primary service the connection status will be set to OK and the toll rates updated template will be returned.

5.26.2.5 CHART2.webservices.tollrateimportmodule:WSTollRateImportModule.initialize (Sequence Diagram)

This diagram shows the processing that is performed by the WSTollRateImportModule during initialization. First a WSVectorImportModuleProperties object is created. Then the TollRateUpdateRequestHandler is created and registered with the RequestManager. Then the DiscoverTollDataConsumersCmd discovery command is created and added to the DiscoveryManager. Next the TollDataManager is created. The TollDataManager schedules it periodic timer task to push current toll data to TollDataConsumers and then depersists its current toll rate data (if any) from a flat file. The TollDataManager is activated in the persistent POA and is registered in the CORBA trading service as both a TollDataConsumer and TollDataProvider. Finally the properties file is read to determine what toll data providers are expected to post data to this service and how frequently they should post their data. For each ClientUpdateAgreement found an ExternalSystemConnection objects is created, activated in the POA and registered in the trading service.

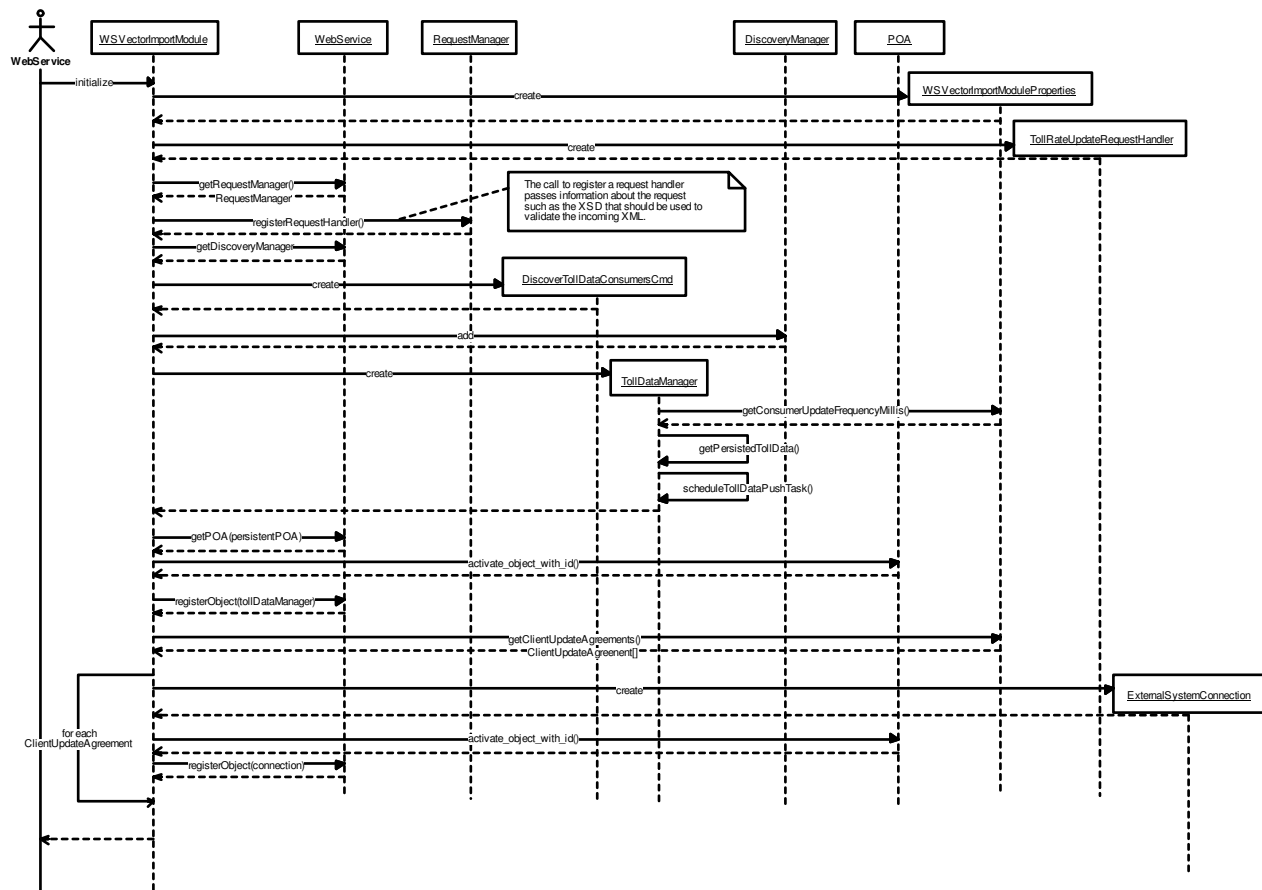


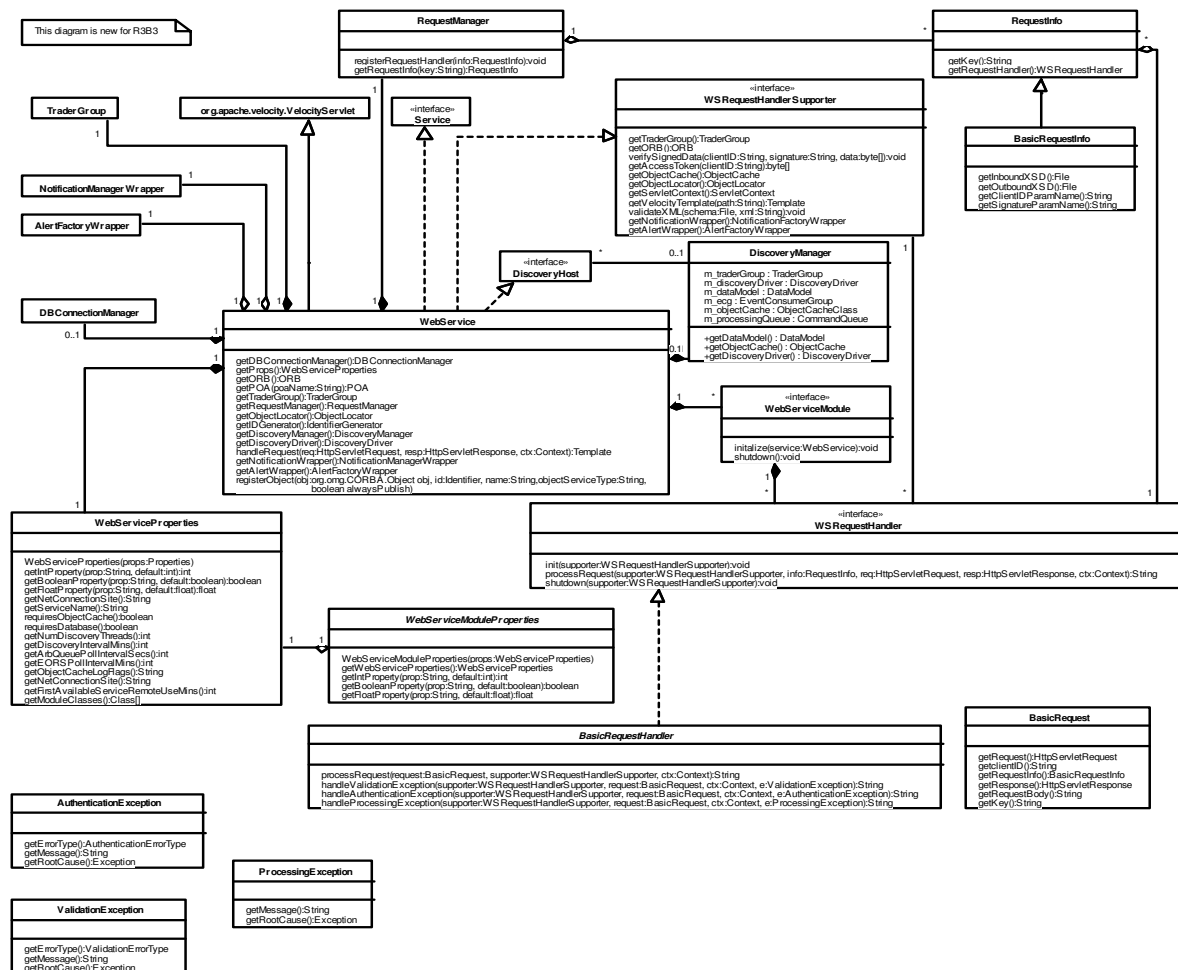
Figure 5-287.
CHART2.webservices.tollrateimportmodule:WSTollRateImportModule.initialize
(Sequence Diagram)

5.27 Webservices.base

5.27.1 Classes

5.27.1.1 WebServicesBaseClasses (Class Diagram)

This diagram shows the classes and interfaces that comprise the Web Service framework. Each WebService reads a properties file to determine which implementations of the WebServiceModule interface should be created and initialized at startup. Each implementation of the WebServiceModule interface can create implementations of the WSRequestHandler interface and install them into the RequestManager.



5-288. WebServicesBaseClasses (Class Diagram)

5.27.1.1.1 AlertFactoryWrapper (Class)

This singleton class provides a wrapper for the Alert Factory that provides automatic location of an Alert Factory and automatic re-discovery should the Alert Factory reference return an error. This class also allows for built-in fault tolerance by automatically failing over to a "working" Alert Factory without the user of this class being aware that this being done. In addition, this class defers the discovery of the Alert Factory until its first use, thus eliminating a start-up dependency for modules that use the Alert Factory.

This class delegates all of its method calls to the system AlertFactory using its currently known good reference to an AlertFactory. If the current reference returns a CORBA failure in the delegated call, this class automatically switches to another reference. When there are no good references (as is true the first time the object is used), this class issues a trader query to (re)discover the published Alert Factory objects in the system. During a method call, the trader will be queried at most one time and under normal circumstances, not at all.

5.27.1.1.2 AuthenticationException (Class)

An instance of this class will be provided to a BasicRequestHandler if the handler has requested digital signature verification but an incoming request does not contain a valid signature.

5.27.1.1.3 BasicRequest (Class)

5.27.1.1.4 This class contains data used during request processing.

5.27.1.1.5 BasicRequestHandler (Class)

This abstract base class provides an implementation of the WSRequestHandler.processRequest() method that provides optional XML validation against specified XSD files and optional digital signature verification as well. It is intended to be used by request handlers that plan to take XML in and return XML to the calling client.

5.27.1.1.6 BasicRequestInfo (Class)

This class provides the request specific data that is required by the BasicRequestHandler implementation of the WSRequestHandler interface.

5.27.1.1.7 DBConnectionManager (Class)

This class implements a database connection manager that manages a pool of database connections. Any CHART II system thread requiring database access gets a database connection from the pool of connections maintained by this manager class. The connections are maintained in two separate lists namely, inUseList and freeList. The inUseList contains connections that have already been assigned to a thread. The freeList contains unassigned connections. This class assumes that an appropriate JDBC driver has been loaded either by

using the "jdbc.drivers" system property or by loading it explicitly. The class has a monitor thread that is started by the constructor. This connection monitor thread periodically checks the inuseList to see if there are connections that are owned by dead threads and move such connections to the freeList. The connection monitor thread is started only if a non-zero value is specified for the monitoring time interval in the constructor.

5.27.1.1.8 DiscoveryHost (Class)

This interface defines the methods that the DiscoveryManager relies on. It must be implemented by any class that will create a DiscoveryManager.

5.27.1.1.9 DiscoveryManager (Class)

This SystemContextProvider interface defines some of the functionality required by a class which provides discovery services for CHART services. It is used by both the CHART GUI and the CHART backend services. A class which implements this interface must provide "get" accessor methods for the system profile properties, the data model, and the main processing queue for a service, for instance. It also provides access to the root deployment path and dynamic image path, which is used only by the CHART GUI. For the CHART GUI, this interface is known to be implemented by the MainServlet; for the back end CHART services, this interface is known to be implemented by the Discovery Manager.

5.27.1.1.10 NotificationManagerWrapper (Class)

This singleton class presents the same interface as the NotificationManager, but uses a FirstAvailableOfferWrapper to provide fault tolerant access to the methods.

5.27.1.1.11 org.apache.velocity.VelocityServlet (Class)

The base class for the Velocity template engine. This template engine is used to provide dynamic content from the CHART GUI Servlet. The web pages are code in templates using velocity specific macros. The code in the servlet loads data that will be shown on the page into a velocity Context, and this VelocityServlet class is used to merge the content with the template to create HTML for the browser to display.

5.27.1.1.12 ProcessingException (Class)

An instance of this class will be provided to a BasicRequestHandler if an unexpected exception is encountered during processing.

5.27.1.1.13 RequestInfo (Class)

This class stores information about a request that the framework will handle. WSRequestHandler instances will register RequestInfo objects to provide the framework information about each request they handle. The framework will pass the registered request information back to the registered WSRequestHandler when the request is being processed.

5.27.1.1.14RequestManager (Class)

This class provides a mapping from a request key to all information that a particular request handler has registered for the request.

5.27.1.1.15Service (Class)

This interface is implemented by all services in the system that allow themselves to be shutdown externally. All implementing classes provide a means to be cleanly shutdown and can be pinged to detect if they are alive.

5.27.1.1.16TraderGroup (Class)

This class provides a facade for trader lookups that allows application level code to be unaware of the number of CORBA trading services that the application is using or the details of the linkage between those services.

5.27.1.1.17ValidationException (Class)

An instance of this class will be provided to a BasicRequestHandler if the handler has requested XSD validation of their incoming or outgoing XML and the XML is found to be not valid. It will contain a ValidationErrorType member that will allow the request handler to determine if the invalid XML was inbound or outbound.

5.27.1.1.18WebService (Class)

This class is the core of each Web Service. It extends the VelocityServlet base class and implements the Service CORBA interface so that Web Service servlets can be administered in the same manner as other CHART service applications.

5.27.1.1.19WebServiceModule (Class)

This interface defines the methods that each module must implement in order to run within the web service framework.

5.27.1.1.20WebServiceModuleProperties (Class)

This abstract base class provides a base for WebServiceModule implementation classes to extend in order to get access to their configuration properties.

5.27.1.1.21WebServiceProperties (Class)

This class provides convenient access to the java Properties object that contains configuration data for the web service and its modules.

5.27.1.1.22WSRequestHandler (Class)

This interface defines the methods that every class that wants to handle web service requests must implement.

5.27.1.1.23WSRequestHandlerSupporter (Class)

This interface defines the methods that will be available to every WSRequestHandler when it is invoked by the framework. It defines the services that the framework will make available to the handlers.

5.27.2 Sequence Diagrams

5.27.2.1 CHART2.webservices.base:BasicRequestHandler.processRequest (Sequence Diagram)

This diagram shows the processing that is performed when a BasicRequest is received by the WebService base. The base calls processRequest on the registered BasicRequestHandler. The BasicRequestHandler base class then gets the client id parameter name and signature parameter names from the registered BasicRequestInfo. It then reads the request body from the request input stream and creates the BasicRequest object which will contain all of the request related data. If the request is registered with a client ID and signature parameter name the base class then calls the WSRequestHandlerSupporter to verify that the digital signature provided is correct for the data contained in the request body and the calling client ID. If it is not, the abstract handleAuthorizationException() method is called to allow the derived request handler class to handle the exception. The velocity template returned by the handler is then returned to the web service so that the error may be returned to the calling client. If no verification was required, or the digital signature was correct, the base class processing then obtains the inbound XSD file from the request info and, performs the optional XML validation. If the inbound XML is not valid the abstract handleValidationException method is invoked to allow the derived request handler to format an error response for the caller. The error response is then returned to the WebService and sent to the calling client. If the inbound XML is valid, or XSD validation was not required the base processing then calls the processRequest() method of the derived request handler to allow it to perform request specific processing. If the processRequest() method throws an unexpected exception the base processing will call the abstract handleProcessingException() method to allow the derived handler an opportunity to format the error response and returns it to the WebService base for return to the calling client. Once the abstract processRequest() method returns, the base processing will check if a response Velocity template has been provided for the creation of the outbound XML. If no template has been returned the derived handler is indicating that they have already responded to the client appropriately and no further processing is required by the base, so a null template path is returned to the WebService and no response is sent to the calling client. If a template path has been returned, the base processing will get the template and merge it with the objects in the Velocity context in order to create the outbound response XML. The registered BasicRequestInfo is then checked to see if outbound XML validation is required. If it is, the XML created by the merge operation is validated. If the outbound XML is not valid the abstract handleValidationException() method is invoked to allow the derived handler to create an appropriate response and that response is returned to the WebService for return to the calling client. If the response XML is valid, or if no validation was required, the response XML is then written to the output stream of the HttpServletResponse and null is returned to the WebService to let it know that the response has already been sent to the calling client.

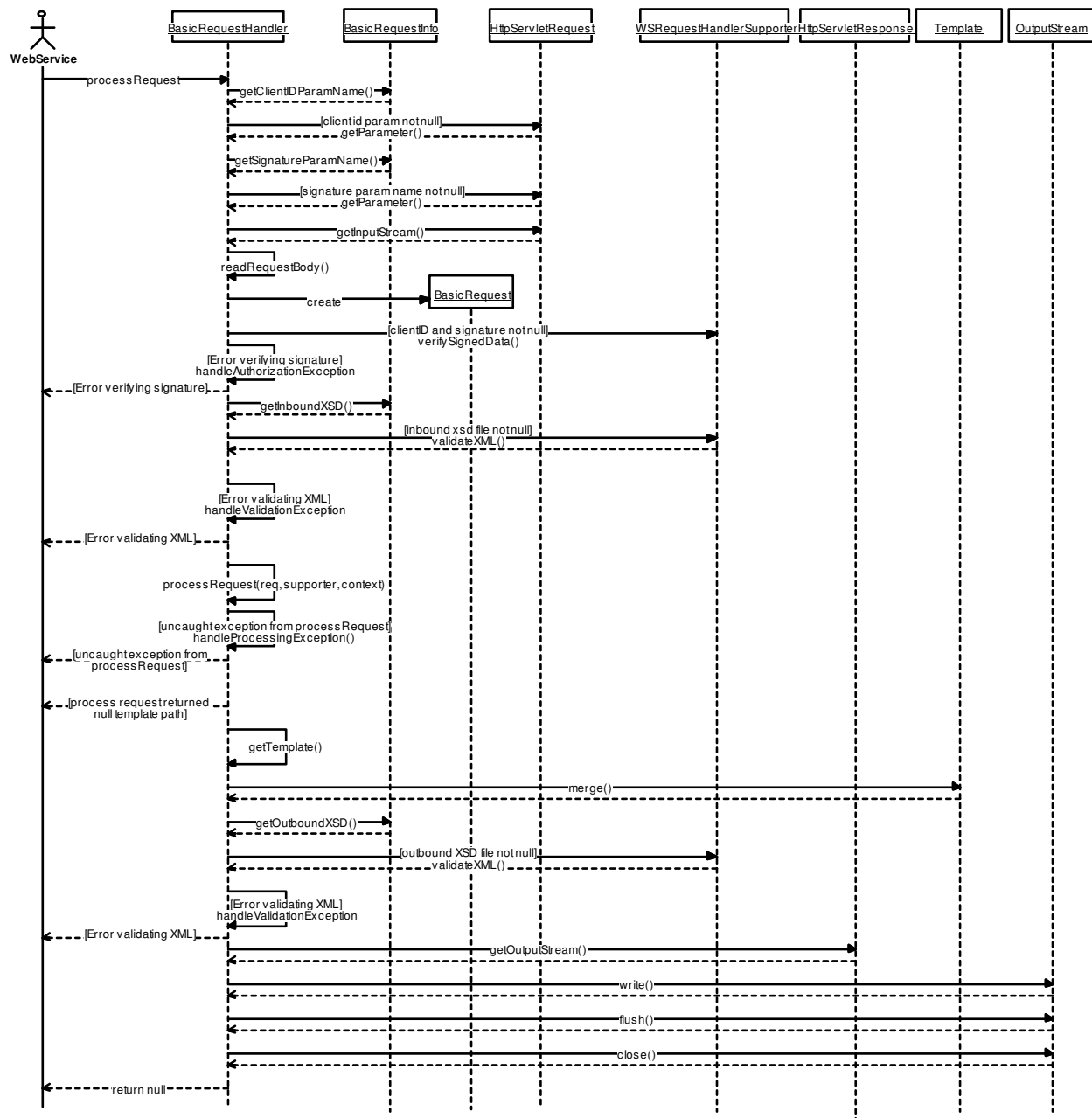


Figure 5-289. CHART2.webservices.base:BasicRequestHandler.processRequest (Sequence Diagram)

5.27.2.2 CHART2.webservices.base:WebService.handleRequest (Sequence Diagram)

This diagram shows the processing performed for each request received by a web service. The request key is determined by parsing the request URL that was received. The key is then used to find the registered RequestInfo which includes the specific implementation of the WSRequestHandler interface that will handle the request. If no RequestInfo can be found about the request key that was received a default XML error response will be returned. If RequestInfo was found, the content type of the response is set to XML and the no-cache header is added to the response. The WSRequestHandler.processRequest() method is then invoked to allow the request handler to perform request specific processing. If the request handler throws an exception the default error XML will be returned to the calling client. Next a check is made to determine if the request handler has already responded to the client. If so, they return null from the process request invocation and the WebService returns null indicating that no response should be sent to the calling client. If the request handler returned a path to a Velocity template that template will be loaded and returned to Velocity where it will be merged with any Java objects placed in the context in order to create the XML response that is returned to the calling client.

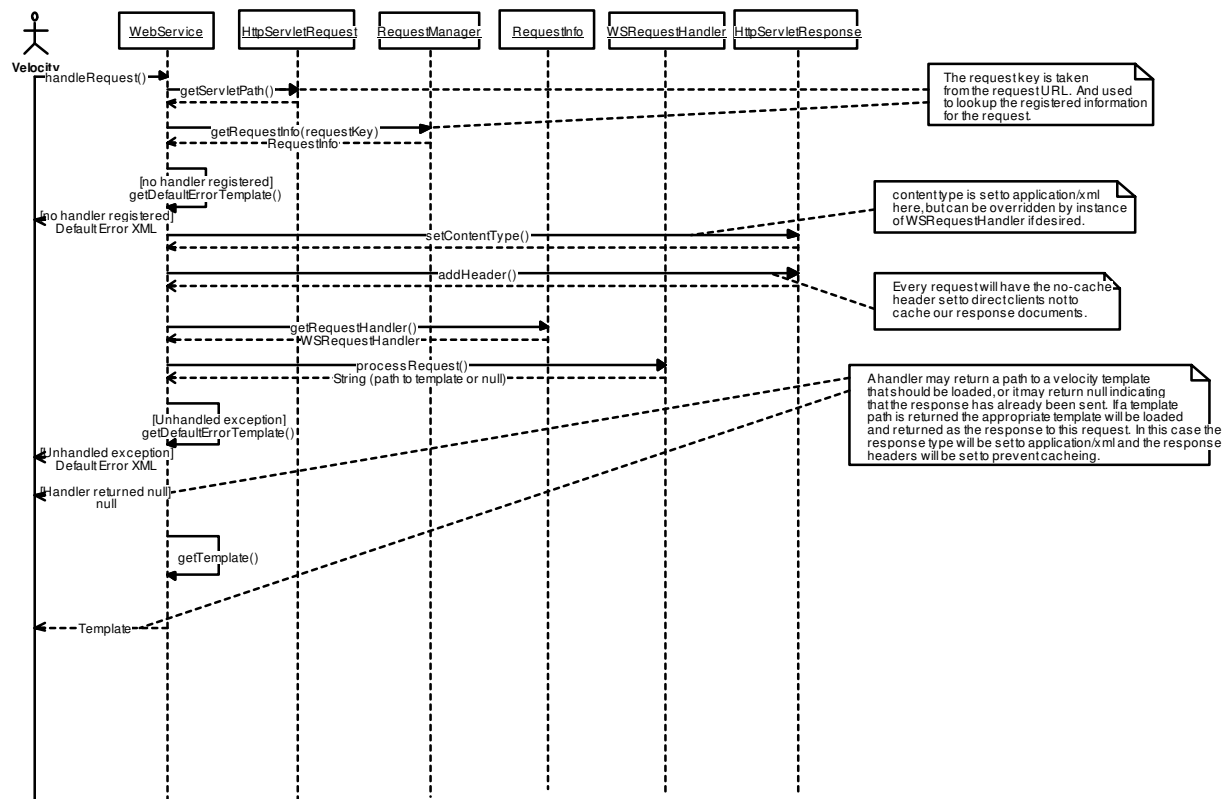


Figure 5-290. CHART2.webservices.base:WebService.handleRequest (Sequence Diagram)

```

sequenceDiagram
    participant SC as ServletContainer
    participant WS as WebService
    participant WSP as WebServiceProperties
    participant DCM as DatabaseConnectionManager
    participant RM as RequestManager
    participant TG as TraderGroup
    participant DM as DiscoveryManager
    participant WSM as WebServiceModule
    participant QC as QueueableCommand
    participant WRH as WSRequestHandler

    SC->>WS: init()
    WS->>WSP: create()
    WSP->>DCM: requiresDatabase()
    WSP->>DCM: (requires db) create()
    WSP->>RM: (requires DB && failure creating connection manager) ServletException
    WSP->>RM: create()
    WSP->>WS: initCORBA()
    WSP->>TG: create()
    WSP->>WS: requiresObjectCache()
    WSP->>DM: (requires object cache) create()
    WSP->>WS: getModuleClasses()
    WSP->>WS: create()
    WSP->>WSM: create()
    WSP->>WSM: for each module class in properties file
    WSP->>WSM: (error constructing module) ServletException
    WSP->>WSM: initialize()
    WSP->>WSM: getDiscoveryManager()
    WSP->>WSM: add()
    WSP->>WSM: getRequestManager()
    WSP->>WSM: create()
    WSP->>WSM: registerRequestHandler()
    WSP->>WSM: for each request a handler handles
    WSP->>WSM: (error initializing module) ServletException
    
```

The module constructed will implement the `WebServiceModule` interface. But will add module specific functionality. During initialization they will be provided the opportunity to register their request handlers and/or add discovery commands to the `DiscoveryManager`.

If the module wants to do discovery and place objects in the `ObjectCache` it will create instances of `QueueableCommand` that do the work and add them to the `DiscoveryManager`.

The `WSRequestHandler` instances that a module installs will be specific to the types of requests that the handler needs to handle. When the request handler is registered it is registered for a specific action. When the servlet receives a request with the specified action the registered handler will be invoked. The method used to invoke the request handler depends on the type of handler registered.

CHART R3B3 Detailed Design

5.28 Chartlite.data

5.28.1 Classes

5.28.1.1 MiscDataClasses (Class Diagram)

This diagram shows miscellaneous classes used by the CHART GUI servlet related to the data cache.



Figure 5-292. MiscDataClasses (Class Diagram)

5.28.1.1.1 BasePushConsumer (Class)

This is a base class for push consumers. Derived classes must implement `handleEventData()`.

5.28.1.1.2 FolderEnabled (Class)

This interface provides access to information about an object that can be stored in a folder.

5.28.1.1.3 NotificationShortcutListItem (Class)

This class represents an item in a notification shortcut list.

5.28.1.1.4 Searchable (Class)

This interface allows objects to be searched for via a substring search.

5.28.1.1.5 SystemProfileNotificationProperties (Class)

This class contains functionality for accessing notification settings in the system profile.

5.28.1.1.6 SystemProfileProperties (Class)

This class is used to cache the system profile properties and provide access to them. It is also used to interact with the server to change system profile settings.

5.28.1.1.7 TempObjectStore (Class)

This class provides a self cleaning storage area for temporary objects.

5.28.1.1.8 WebAdministered (Class)

This interface allows the implementing class to be administered via the trader console pages.

5.28.1.1.9 WebDevice (Class)

This interface contains common functionality for CHART devices.

5.28.1.1.10 WebHARMessageNotifier (Class)

This interface provides access to HAR notification capabilities for a device (DMS or SHAZAM) that is used to notify the public of a HAR message being broadcast.

5.28.1.1.11 WebOpCenter (Class)

This class is used to wrap an `OperationsCenter` object to allow it to be cached in the CHART GUI servlet and to allow the cached data to be accessed within Velocity templates.

5.28.1.1.12WebSharedResource (Class)

This interface is implemented by any GUI-side wrapper objects representing CHART shared resources in the system, corresponding to the SharedResource IDL interface.

5.28.1.1.13WebUniquelyIdentifiable (Class)

This interface provides functionality for GUI objects that represent UniquelyIdentifiable objects as defined in the IDL.

5.28.1.2 MiscDataClasses2 (Class Diagram)

This diagram contains additional classes in the chartlite.data package that did not fit on the first diagram.

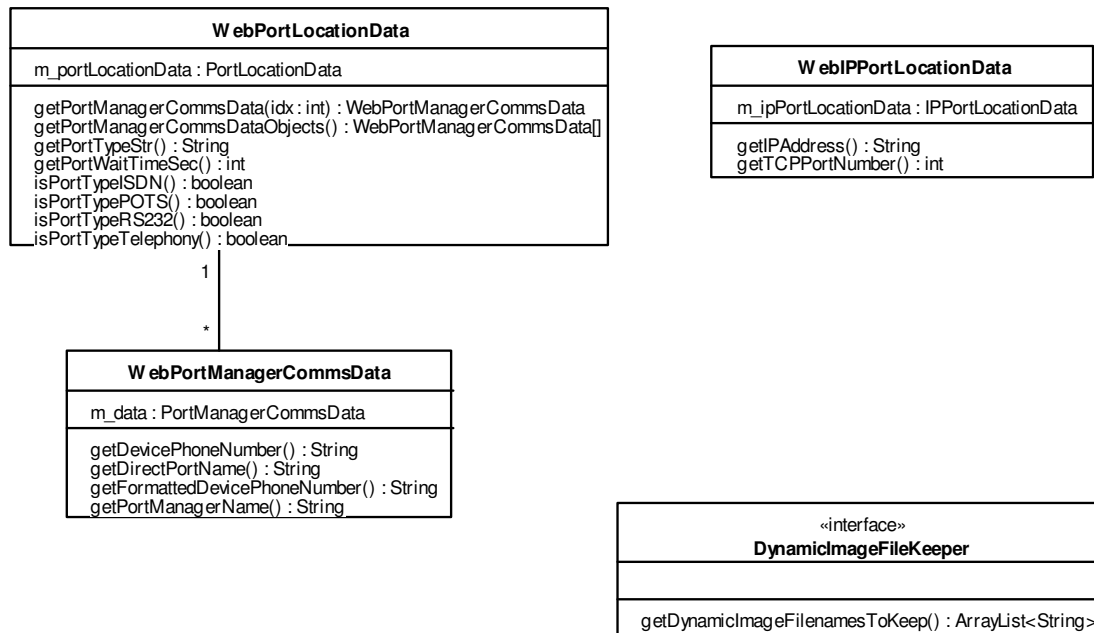


Figure 5-293. MiscDataClasses2 (Class Diagram)

5.28.1.2.1 DynamicImageFileKeeper (Class)

This interface allows an object to keep dynamic image files from being deleted by the `DynImageCleanupTask`, which periodically deletes files that are no longer needed.

5.28.1.2.2 WebIPPortLocationData (Class)

This class wraps the `IPPortLocationData` IDL structure and provides accessor methods to get the data. This class has data for identifying a TCP/IP address and port.

5.28.1.2.3 WebPortLocationData (Class)

This class wraps the `PortLocationData` IDL structure and provides accessors to get the data. The structure contains data for using ports of the same type, managed by one or more `PortManager` objects.

5.28.1.2.4 WebPortManagerCommsData (Class)

This class wraps the `PortManagerCommsData` IDL structure and provides accessors to get the data. The structure contains data for using a port managed by a `PortManager` object.

5.28.1.3 chartlite.data_location_classes (Class Diagram)

This diagram shows classes used by the CHART GUI servlet related to location information that is cached in the data model.

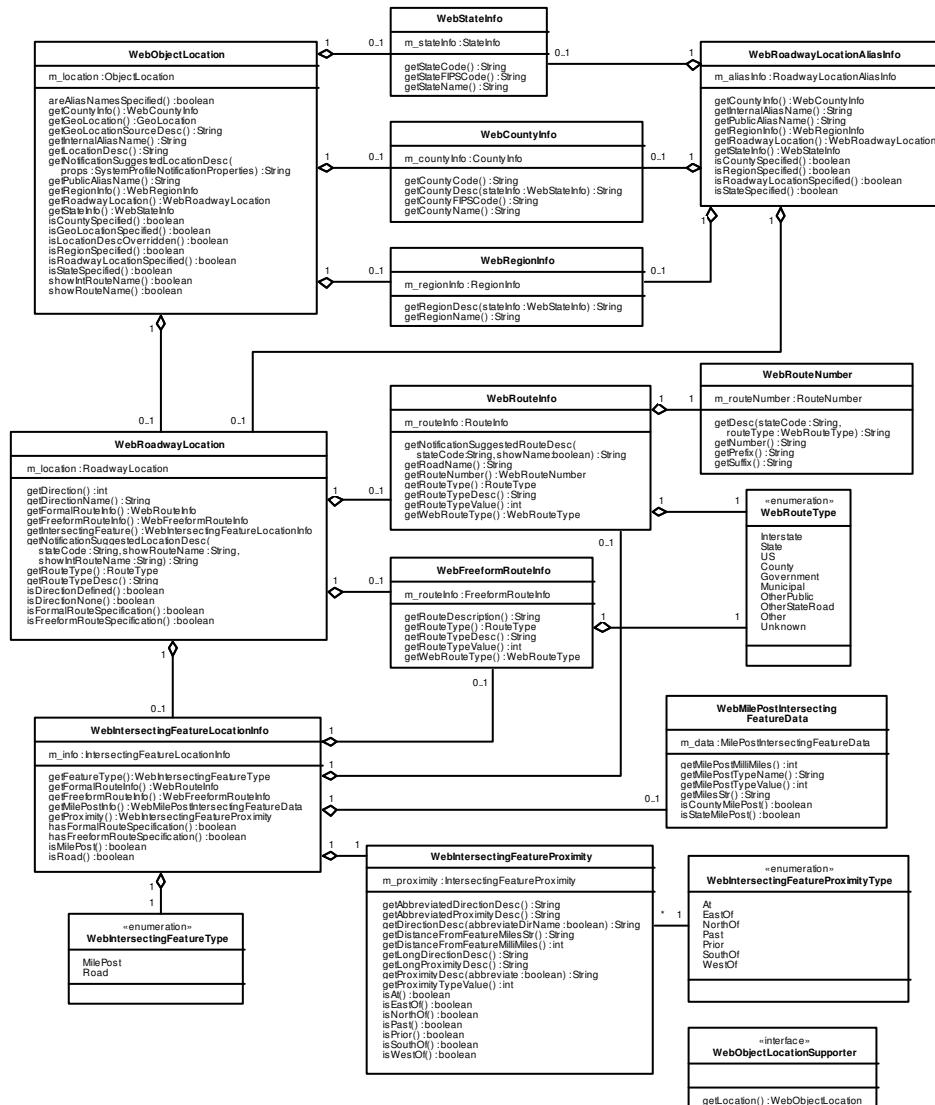


Figure 5-294. chartlite.data_location_classes (Class Diagram)

5.28.1.3.1 WebCountyInfo (Class)

This class provides access to the CountyInfo struct which contains information about a county.

5.28.1.3.2 WebFreeformRouteInfo (Class)

This class provides access to the FreeformRouteInfo struct which contains information

about a route where only the route description is known (not the formal route number).

5.28.1.3.3 WebIntersectingFeatureLocationInfo (Class)

This class provides access to the IntersectingFeatureLocationInfo struct which contains information about a point along a roadway.

5.28.1.3.4 WebIntersectingFeatureProximity (Class)

This class provides access to the IntersectingFeatureProximity struct which contains information about the proximity (direction and distance) of a point relative to an intersecting feature.

5.28.1.3.5 WebIntersectingFeatureProximityType (Class)

The enumeration contains the supported proximity values to describe a point relative to an intersecting feature.

5.28.1.3.6 WebIntersectingFeatureType (Class)

The enumeration contains the supported types of intersecting features.

5.28.1.3.7 WebMilePostIntersecting FeatureData (Class)

This class contains data describing a mile post intersecting feature, which can be a state or county milepost.

5.28.1.3.8 WebObjectLocation (Class)

This class provides access to the ObjectLocation struct which contains information about the location of an object in the system.

5.28.1.3.9 WebObjectLocationSupporter (Class)

This interface allows common processing for objects supporting an ObjectLocation via the WebObjectLocation wrapper class..

5.28.1.3.10WebRegionInfo (Class)

This class provides access to the RegionInfo struct which contains information about a region.

5.28.1.3.11WebRoadwayLocation (Class)

This class provides access to the RoadwayLocation struct which contains information about a location on a roadway.

5.28.1.3.12WebRoadwayLocationAliasInfo (Class)

This class provides access to the RoadwayLocationAliasInfo struct which contains

information about a location alias.

5.28.1.3.13 WebRouteInfo (Class)

This class provides access to the RoutInfo struct which contains information about a formal route specification (i.e., one where the route prefix, number, and suffix are used).

5.28.1.3.14 WebRouteNumber (Class)

This class provides access to the RouteNumber struct which contains information about a formal route number, where the prefix, number, and suffix are known.

5.28.1.3.15 WebRouteType (Class)

This enumeration contains the allowable route types.

5.28.1.3.16 WebStateInfo (Class)

This class provides access to the StateInfo struct which contains information about a state.

5.29.1.1.1 ArbitratedDevice (Class)

This interface allows a class to use a WebArbQueue to track the current state of a device's arbitration queue.

5.29.1.1.2 DMSTravInfoMsg DataSupplier (Class)

This interface provides data for travel routes used in a DMSTravInfoMsg. It will be used to substitute the template tags with route-specific data, in order to format the template and produce MULTI. This is needed in the GUI for true display, and is needed in the server for formatting messages to send to a DMS. The routeNum parameter corresponds to route numbers contained in the template data tags, and it is a 1-based index. These methods will throw an exception if the requested data is not available.

5.29.1.1.3 DMSTravInfoMsgTrueDisplayMgr (Class)

This class manages the true display image for a single DMS traveler info message.

5.29.1.1.4 DynamicImage FileKeeper (Class)

This interface allows an object to keep dynamic image files from being deleted by the DynImageCleanupTask, which periodically deletes files that are no longer needed.

5.29.1.1.5 FolderEnabled (Class)

This interface provides access to information about an object that can be stored in a folder.

5.29.1.1.6 ModelObserver (Class)

This interface must be implemented by any object which would like to attach to the DataModel as an observer and get updated as system objects are added, deleted or changed.

5.29.1.1.7 NameFilterable (Class)

This java interface is implemented by classes which can be filter by name within the ObjectCache. A NameFilter object is passed into the ObjectCache to select NameFilterable objects in the cache.

5.29.1.1.8 Searchable (Class)

This interface allows objects to be searched for via a substring search.

5.29.1.1.9 WebAdministered (Class)

This interface allows the implementing class to be administered via the trader console pages.

5.29.1.1.10WebChart2DMS (Class)

This class extends WebDMS and wraps the Chart2DMS CORBA interface, providing

access to CHART2-specific functionality.

5.29.1.1.11 WebChart2DMSConfiguration (Class)

This class wraps the Chart2DMSConfiguration IDL valuetype and adds accessor methods.

5.29.1.1.12 WebDevice (Class)

This interface contains common functionality for CHART devices.

5.29.1.1.13 WebDMS (Class)

This class represents a dynamic message sign.

5.29.1.1.14 WebDMSConfiguration (Class)

This class wraps the DMSConfiguration IDL structure and adds accessor methods.

5.29.1.1.15 WebDMSTravInfoMsg (Class)

This class wraps the DMSTravInfoMsg IDL structure that represents a traveler info message used by a DMS, and provides accessor methods.

5.29.1.1.16 WebExternalDMS (Class)

This class wraps the ExternalDMS CORBA interface and provides access to cached data specific to external DMSs.

5.29.1.1.17 WebExternalDMS Configuration (Class)

This class wraps the ExternalDMSConfiguration IDL structure and provides accessor methods to access the data.

5.29.1.1.18 WebHARMessageNotifier (Class)

This interface provides access to HAR notification capabilities for a device (DMS or SHAZAM) that is used to notify the public of a HAR message being broadcast.

5.29.1.1.19 WebHHMMRange (Class)

This class contains information about a time-of-day range that contains hours and minutes.

5.29.1.1.20 WebObjectLocationSupporter (Class)

This interface allows common processing for objects supporting an ObjectLocation via the WebObjectLocation wrapper class..

5.29.1.1.21 WebSharedResource (Class)

This interface is implemented by any GUI-side wrapper objects representing CHART shared resources in the system, corresponding to the SharedResource IDL interface.

5.29.1.2 GUIDMSDataClasses2 (Class Diagram)

This diagram shows additional GUI data classes related to DMS management.

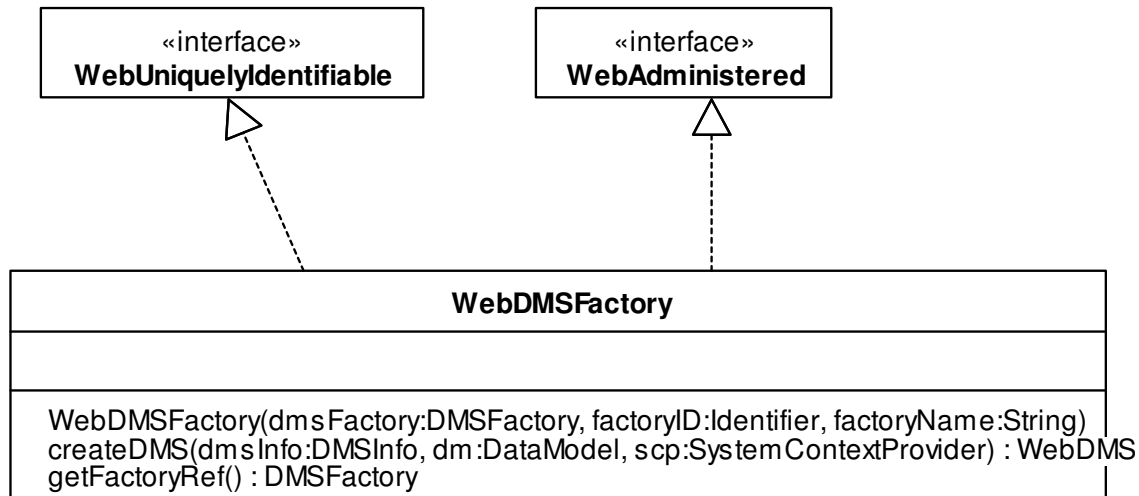


Figure 5-296. GUIDMSDataClasses2 (Class Diagram)

5.29.1.2.1 WebAdministered (Class)

This interface allows the implementing class to be administered via the trader console pages.

5.29.1.2.2 WebDMSFactory (Class)

This class wraps the DMSFactory CORBA interface and provides additional GUI functionality including caching the factory name and ID.

5.29.1.2.3 WebUniquelyIdentifiable (Class)

This interface provides functionality for GUI objects that represent UniquelyIdentifiable objects as defined in the IDL.

5.29.2 Sequence Diagrams

5.29.2.1 DiscoverDMSClassesCommand:discoverDMSClasses (Sequence Diagram)

This diagram shows the processing to find (discover) DMS-related objects, which happens periodically or as requested. The DMS factories are queried from the trader group, and if the WebDMSFactory objects are not already found in the DataModel cache, the wrapper objects are created and added to the cache. Each factory is then called to obtain the list of DMSs it owns. If the corresponding WebDMS is already in the DataModel cache, it is retrieved from the cache and is called to update its cached configuration and status data. If not already in the cache, the WebDMSFactory class is called to create the new WebDMS wrapper object, as shown in the WebDMSFactory:createDMS sequence diagram.

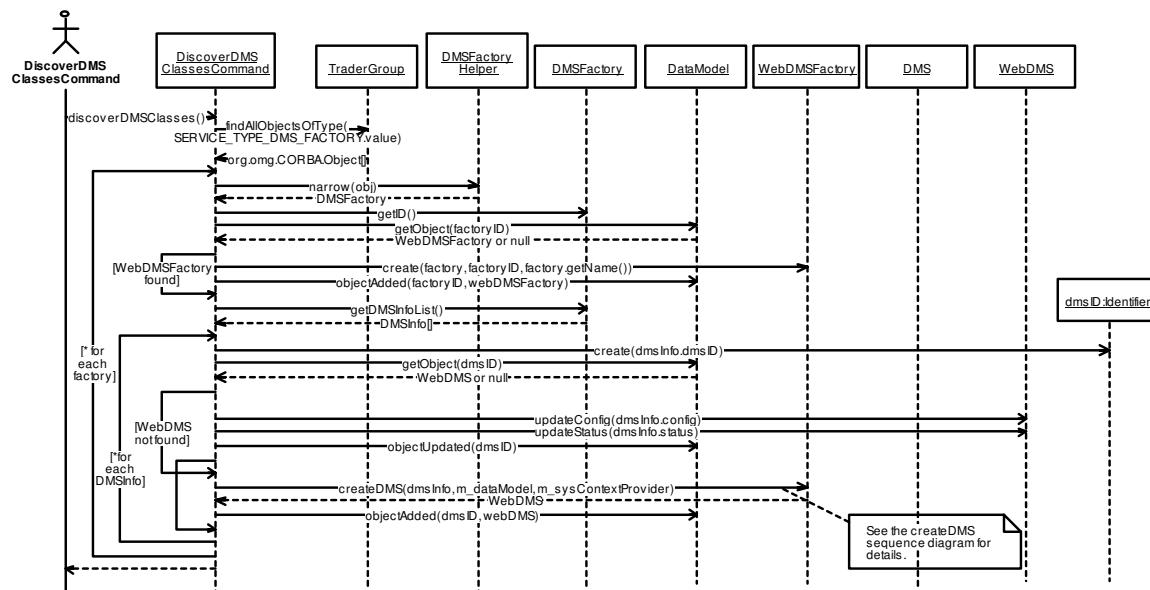


Figure 5-297. DiscoverDMSClassesCommand:discoverDMSClasses (Sequence Diagram)

5.29.2.2 DMSTravInfoMsgTrueDisplayMgr:updateGIF (Sequence Diagram)

This diagram shows how the graphical representation of a DMS traveler info message is updated. The DMSTravInfoMsgTrueDisplayMgr is called to update the image. It gets the MultiMsgGIFEncoder from the WebDMS, which can be null if the sign is of the wrong type or the font is not available. The template ID is used to retrieve the cached WebDMSTravInfoMsgTemplate, to get the template configuration. Next a DMSTravInfoMsgTemplateFormatter is created and called to format the template, replacing the template's tags to obtain a MULTI representation of the message using the current travel route data. After getting the MULTI, the MultiMsgGIFEncoder is called to encode the message as a GIF image. The GIF filename and pixel dimensions are stored for later use.

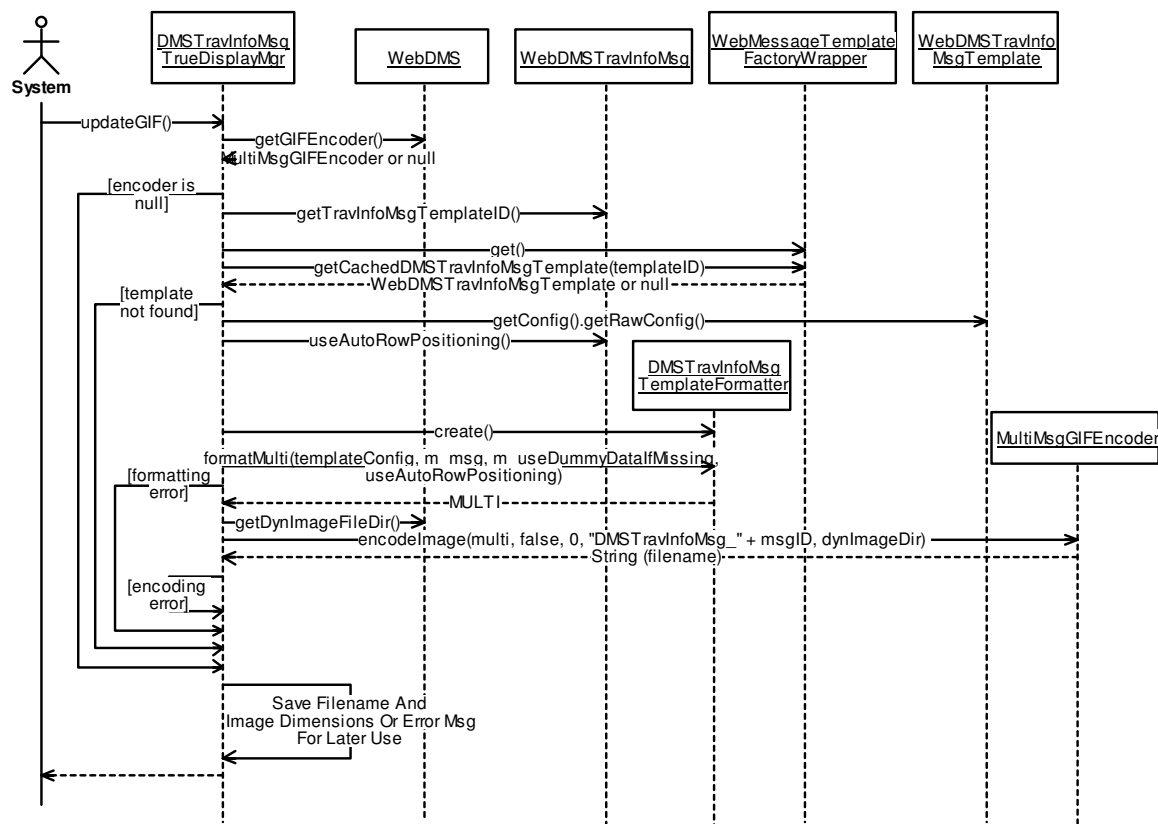


Figure 5-298. DMSTravInfoMsgTrueDisplayMgr:updateGIF (Sequence Diagram)

5.29.2.3 WebChart2DMS:create (Sequence Diagram)

This diagram shows the processing when a new WebChart2DMS wrapper is created. First it calls the WebDMS base class to initialize, which sets up the GIF encoder. The WebChart2DMS then constructs a WebArbQueue object, and sets up the WebDMSTravInfoMsg objects and initializes the corresponding GIF files as described in the sequence diagram: setupDMSTravInfoMsgs.

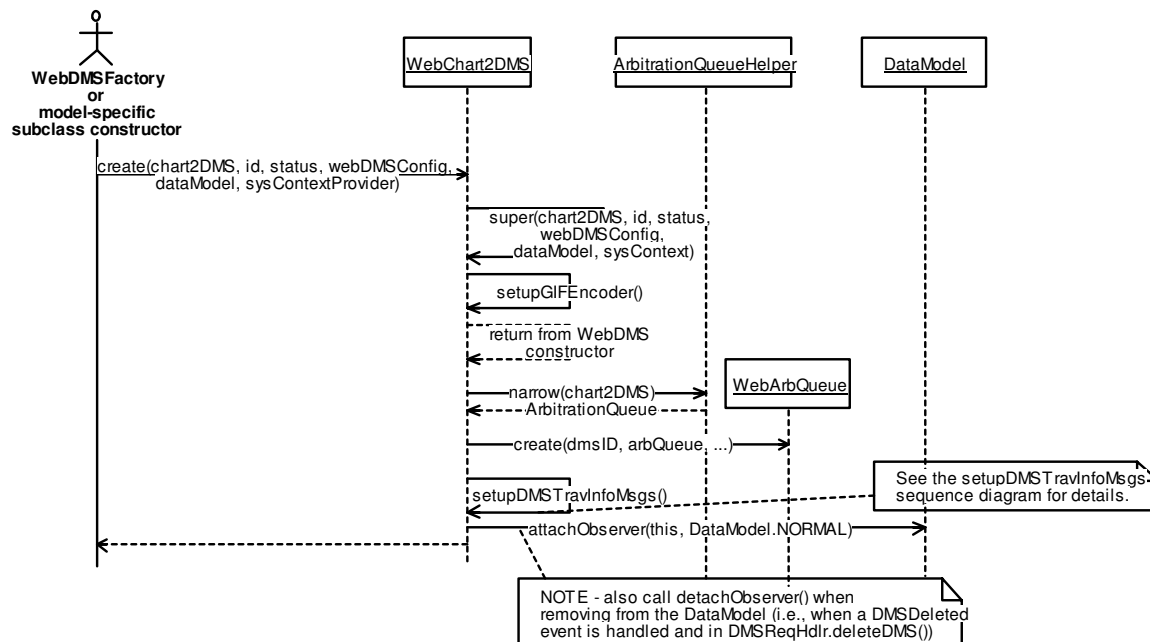


Figure 5-299. WebChart2DMS:create (Sequence Diagram)

5.29.2.4 WebChart2DMS:setupDMSTravInfoMsgs (Sequence Diagram)

This diagram shows how the DMS traveler info messages are set up within the WebChart2DMS wrapper object. For each DMSTravInfoMsg within the configuration, it creates a new WebDMSTravInfoMsg wrapper and associated DMSTravInfoMsgTrueDisplayMgr for managing the image files. The GIF image for each message is created, as described in the updateGIF sequence diagram.

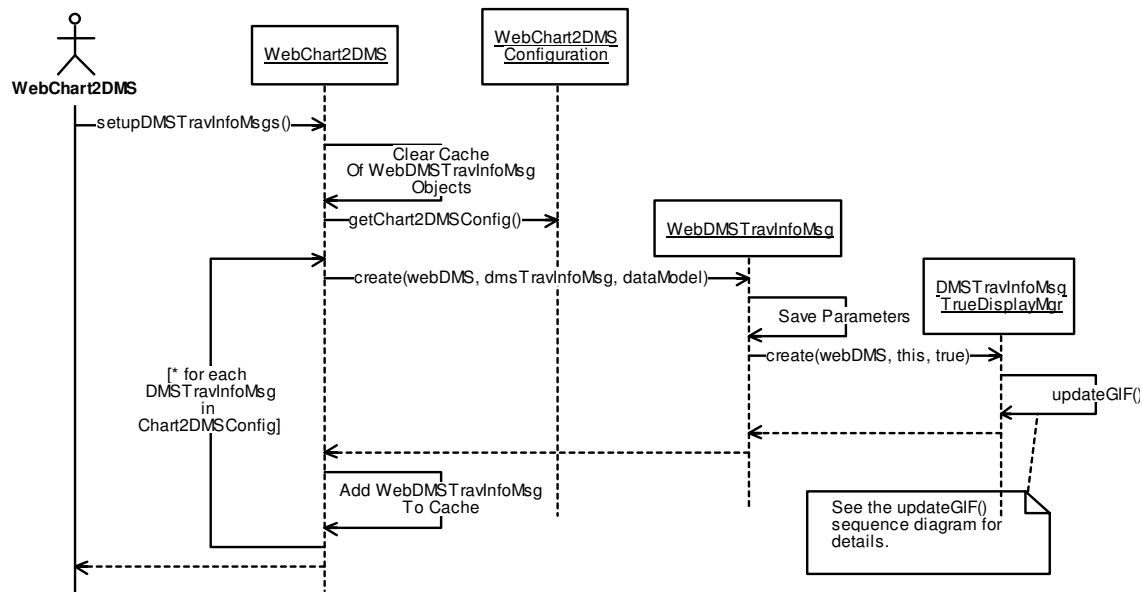


Figure 5-300. WebChart2DMS:setupDMSTravInfoMsgs (Sequence Diagram)

5.29.2.5 WebChart2DMS:updateConfig (Sequence Diagram)

This diagram shows the processing when the CHART2DMSConfiguration is updated, which can happen during discovery or as a result of CORBA events being pushed. If the CORBA event param is null or the event type indicates a config change, the WebArbQueue is updated, the entire Chart2DMSConfiguration is replaced within the WebChart2DMSConfiguration wrapper object, and the GIF encoder is set up again. If the event is null, all of the WebDMSTravInfoMsg objects are replaced and their GIF files are regenerated. If these objects are not replaced but the GIF encoder was updated, the GIFs managed by all of the WebDMSTravInfoMsg objects are updated. If the event type indicates a DMSTravInfoMsg was added, a new WebDMSTravInfoMsg is created and added to the cache. If a message was updated, the WebDMSTravInfoMsg is called to update itself and update the GIF file it manages. If a message was removed, the corresponding WebDMSTravInfoMsg is removed from the cache. Finally if a message was added, changed, or removed, the array of DMSTravInfoMsg within the Chart2DMSConfiguration is replaced with the array from the new configuration.

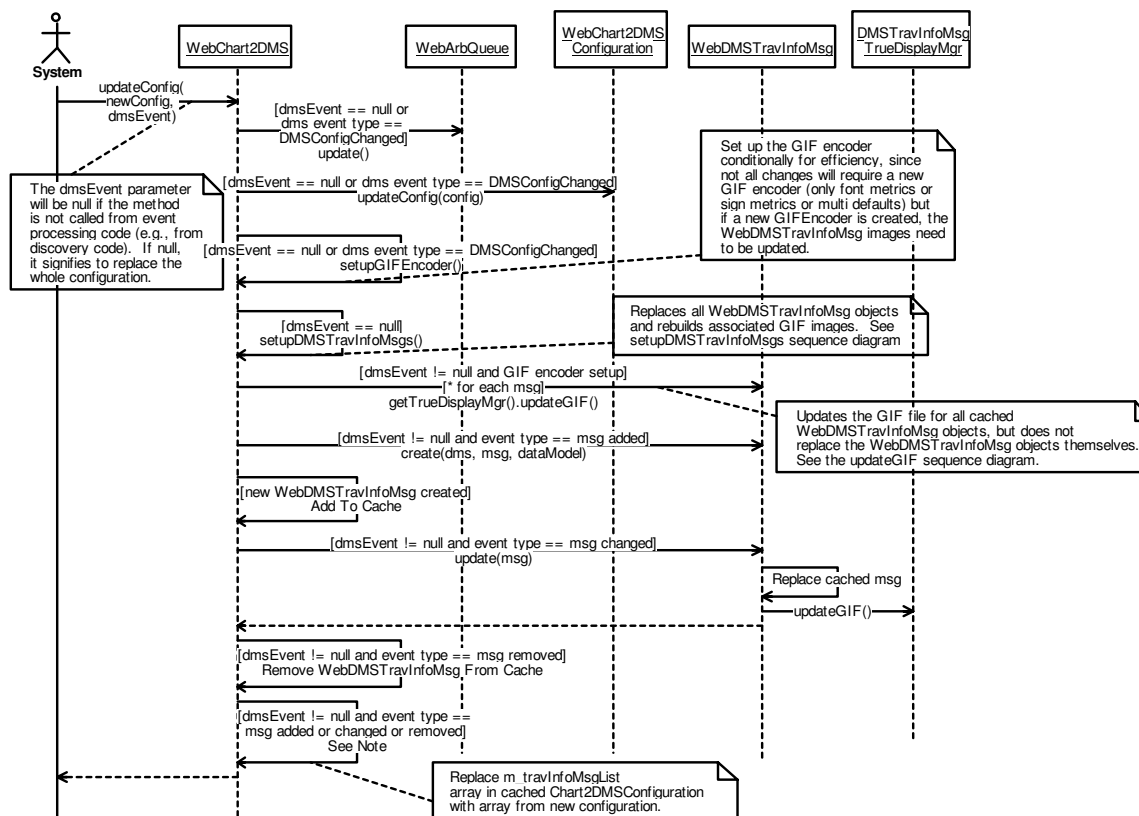


Figure 5-301. WebChart2DMS:updateConfig (Sequence Diagram)

5.29.2.6 WebChart2DMS:update_ModelChange (Sequence Diagram)

This diagram shows the processing when the WebChart2DMS is called by the DataModel when object changes have occurred. Only changes related to the WebTravelRoute class are considered, and the ID of a changed WebTravelRoute is found and all of the cached WebDMSTravInfoMsg objects are asked whether they are using the route. If the message is using the route, its ID is put into a HashSet to mark those messages needing image updates. After all changed routes have been processed, the message IDs stored in the HashSet are used to get the WebDMSTravInfoMsg objects and update their images. (The use of the DataModel observer mechanism and the HashSet prevents an image from being generated if it contains multiple routes with changed data, as multiple route changes are accumulated and processed at one time.)

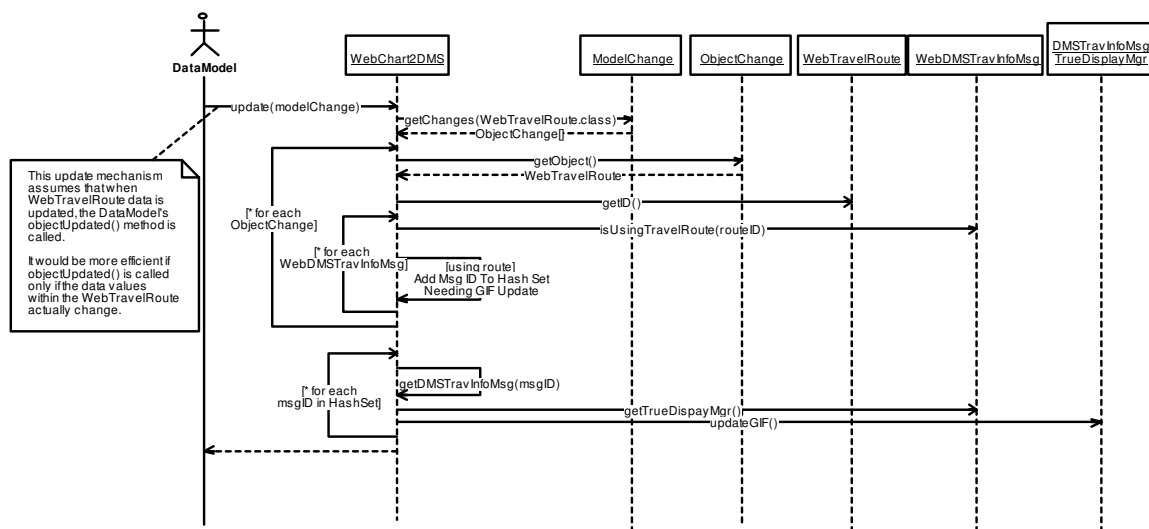


Figure 5-302. WebChart2DMS:update_ModelChange (Sequence Diagram)

5.29.2.7 WebDMSFactory:createDMS (Sequence Diagram)

This diagram shows how a WebDMS object is created, or one of its subclasses. The dmsType is used to determine whether the DMS is an external DMS or a CHART DMS. If it is a CHART DMS, the model ID from the Chart2DMSConfiguration is used to determine what model of DMS is used. Depending on these values the CORBA "helper" classes are called to get the type-specific CORBA interface references: ExternalDMS, FP9500DMS, TS3001DMS, SylviaDMS, or PCMSDMS. Then these references are used to create the appropriate type-specific wrapper object, after creating a WebChart2DMSConfiguration to wrap the configuration. If it is a CHART DMS but not one the above types, a WebChart2DMS is created. If it is not a CHART or external DMS, a WebDMS is created.

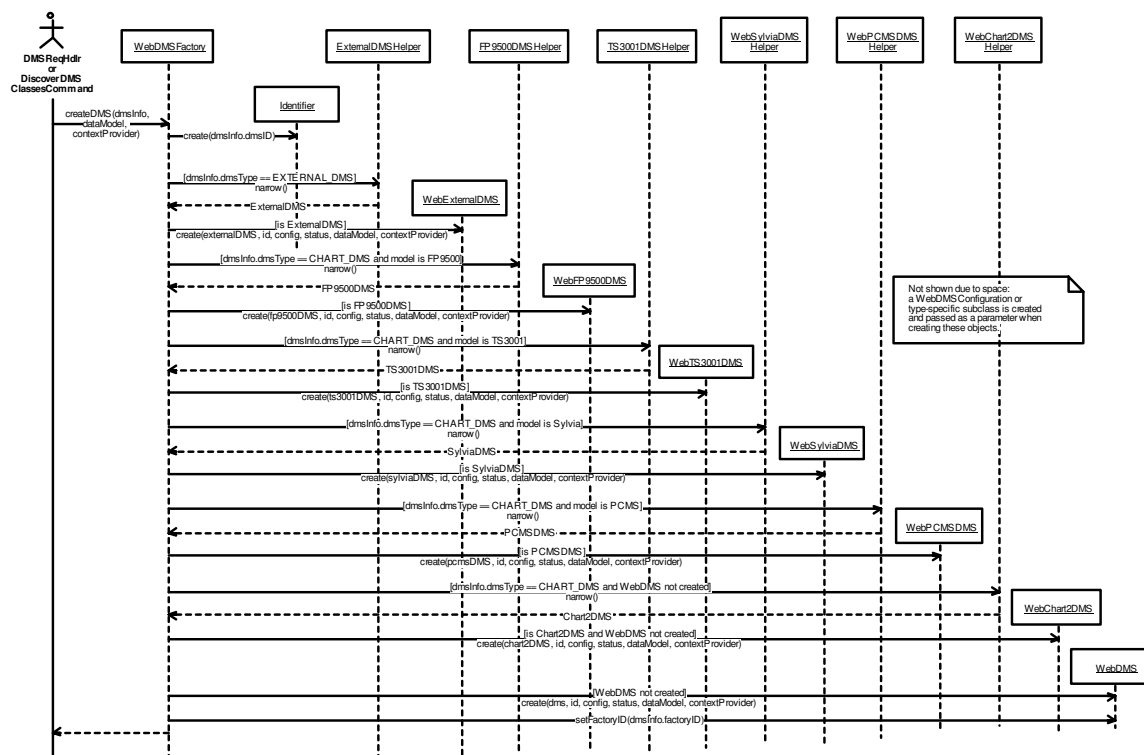


Figure 5-303. WebDMSFactory:createDMS (Sequence Diagram)

5.30 Chartlite.data.video-data

5.30.1 Classes

5.30.1.1 GUIVideoDataClasses (Class Diagram)

This diagram shows GUI data classes related to video management.

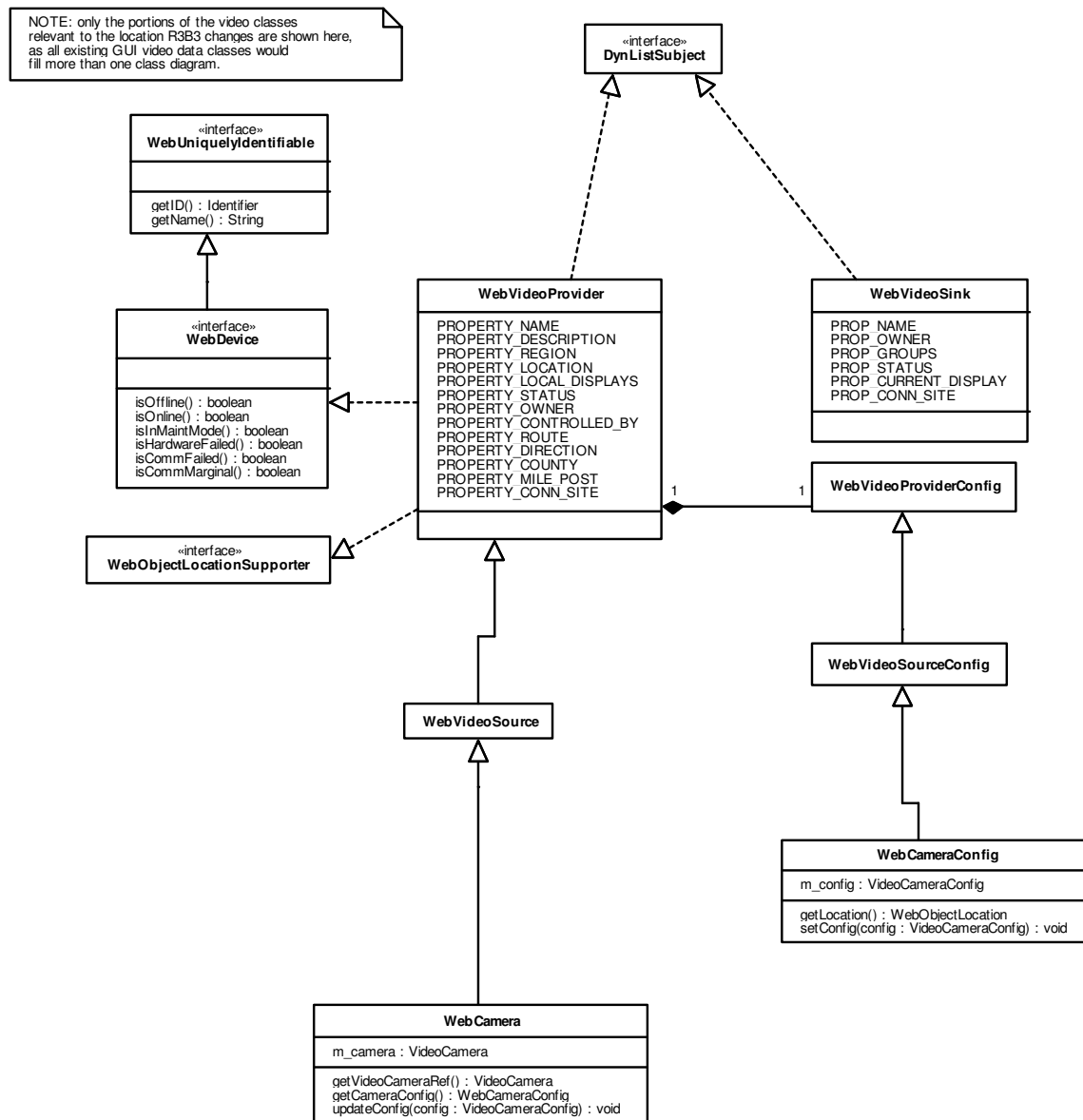


Figure 5-304. GUIVideoDataClasses (Class Diagram)

5.30.1.1.1 DynListSubject (Class)

This interface is implemented by classes that wish to be capable of being displayed in a dynamic list.

5.30.1.1.2 WebCamera (Class)

This class wraps the VideoCamera CORBA reference and stores cached configuration and status for fast local access.

5.30.1.1.3 WebCameraConfig (Class)

This class wraps the VideoCameraConfig structure defined in the IDL and provides accessor methods.

5.30.1.1.4 WebDevice (Class)

This interface contains common functionality for CHART devices.

5.30.1.1.5 WebObjectLocationSupporter (Class)

This interface allows common processing for objects supporting an ObjectLocation via the WebObjectLocation wrapper class..

5.30.1.1.6 WebUniquelyIdentifiable (Class)

This interface provides functionality for GUI objects that represent UniquelyIdentifiable objects as defined in the IDL.

5.30.1.1.7 WebVideoProvider (Class)

This class wraps the VideoProvider CORBA reference and stores cached configuration and status for fast local access.

5.30.1.1.8 WebVideoProviderConfig (Class)

This class wraps the VideoProviderConfig structure defined in the IDL and provides accessor methods.

5.30.1.1.9 WebVideoSink (Class)

This class wraps the VideoSink CORBA reference and stores cached configuration and status for fast local access.

5.30.1.1.10 WebVideoSource (Class)

This class wraps the VideoSource CORBA reference and stores cached configuration and status for fast local access.

5.30.1.1.11 WebVideoSourceConfig (Class)

This class wraps the VideoSourceConfig structure defined in the IDL and provides accessor methods.

5.31 Chartlite.data.location-data

5.31.1 Classes

5.31.1.1 GUILocationDataClasses (Class Diagram)

This diagram shows data classes related to location data. Most location classes can be found in the parent data package.



Figure 5-305. GUILocationDataClasses (Class Diagram)

5.31.1.1.1 WebRoadwayLocationLookup (Class)

This class wraps the RoadwayLocationLookup interface and provides default values, data caching, and other auxiliary functionality.

5.32 Chartlite.data.shazam-data

5.32.1 Classes

5.32.1.1 GUIShazamClasses (Class Diagram)

This diagram shows classes related to SHAZAMs that are used to store data pertaining to SHAZAMs in the GUI object cache.

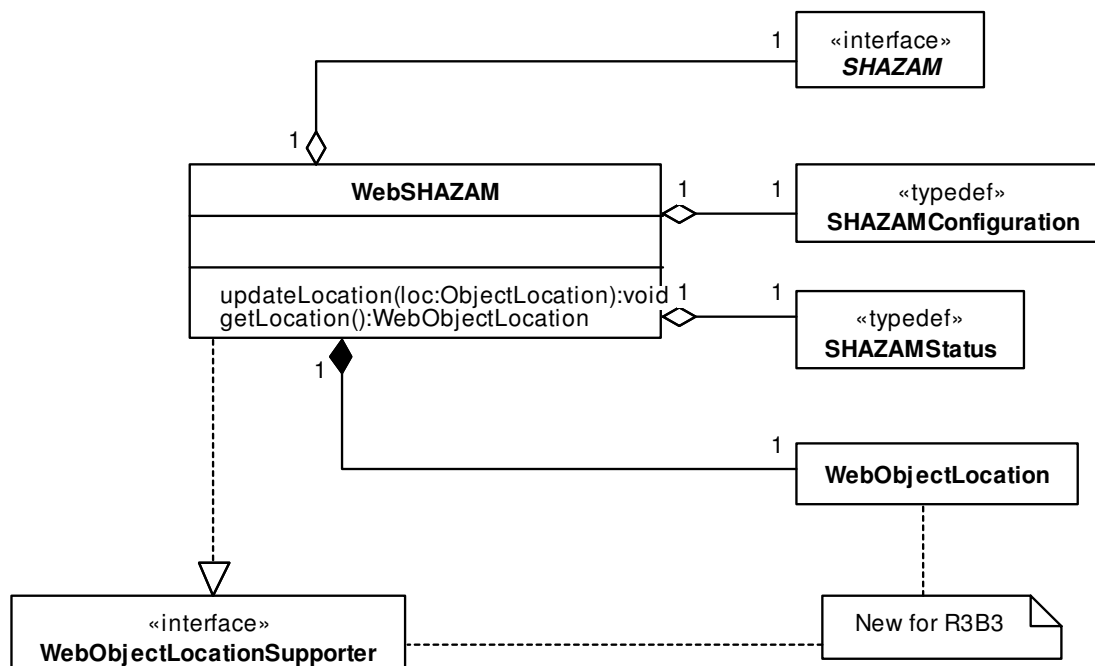


Figure 5-306. GUIShazamClasses (Class Diagram)

5.32.1.1.1 SHAZAM (Class)

This interface class is used to identify the SHAZAM-specific methods which can be used to interface with a SHAZAM field device. It specifies methods for activating and deactivating the SHAZAM in maintenance mode, refreshing the SHAZAM (commanding the device to its last known status), changing the configuration of the SHAZAM, and removing the SHAZAM. This interface is implemented by a **SHAZAMImpl** class, which uses a helper **ProtocolHdlr** class to perform the model specific protocol for device command and control.

5.32.1.1.2 SHAZAMConfiguration (Class)

This class contains data that specifies the configuration of a SHAZAM device. It is used to

communicate configuration information to/from the database, and to/from the GUI clients. The GUI sends a SHAZAMConfiguration when creating a SHAZAM or modifying the configuration of an existing SHAZAM. Device Location member has been modified for R3B3. Now it contains a detailed location information.

5.32.1.1.3 SHAZAMStatus (Class)

This class contains the current status of a SHAZAM device. This class is used to store status within the SHAZAM object, and is also used to communicate configuration information to/from the database, and to the GUI clients (one-way).

5.32.1.1.4 WebObjectLocation (Class)

This class provides access to the ObjectLocation struct which contains information about the location of an object in the system.

5.32.1.1.5 WebObjectLocationSupporter (Class)

This interface allows common processing for objects supporting an ObjectLocation via the WebObjectLocation wrapper class..

5.32.1.1.6 WebSHAZAM (Class)

This class is a wrapper for a SHAZAM CORBA object, used to cache data related to the SHAZM in the GUI object cache and to provide access to the SHAZAM configuration and status data on web pages.

5.33 Chartlite.data.har-data

5.33.1 Classes

5.33.1.1 GUIHARDataClasses (Class Diagram)

This diagram shows classes used to store HAR related data in the GUI cache. New for R3B3 is the `WebObjectLocation` which is available in a `WebHARConfig`.

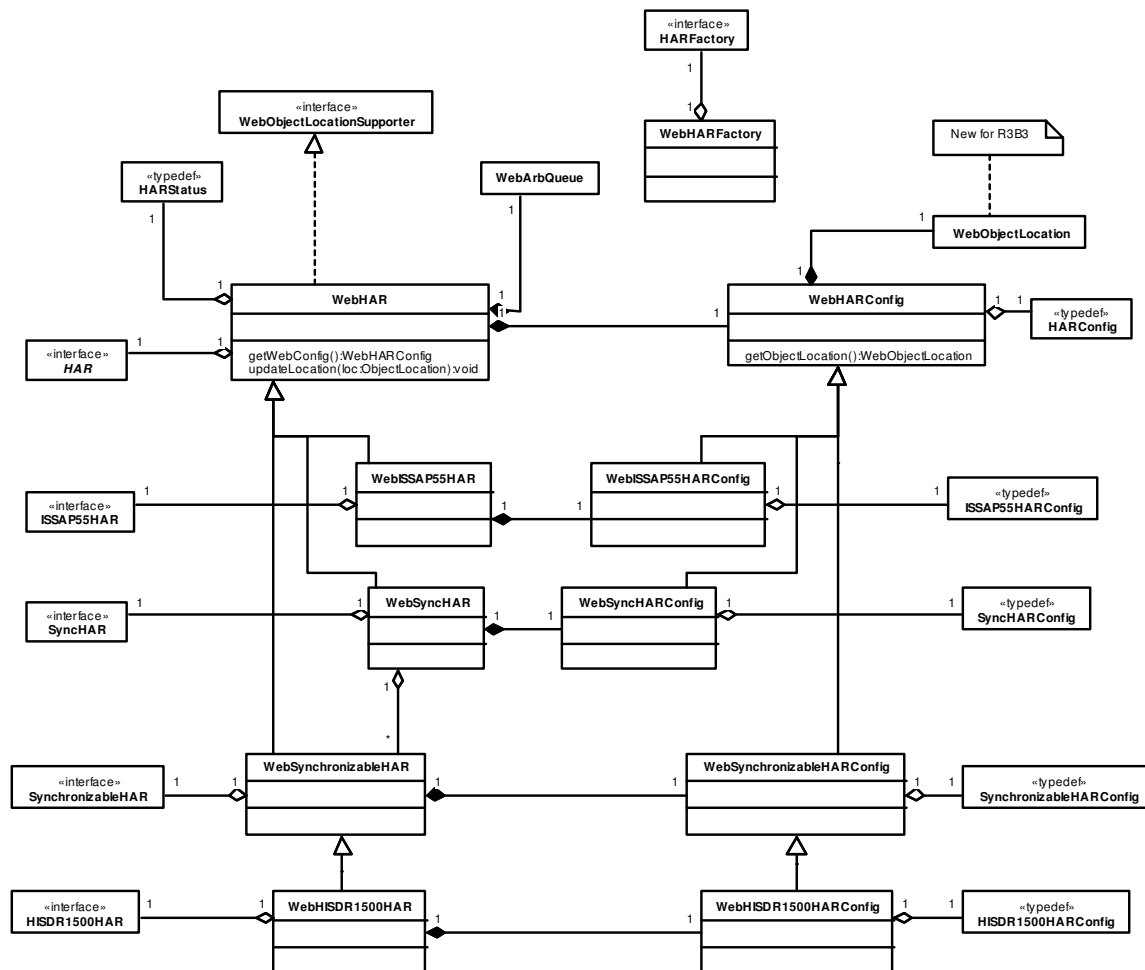


Figure 5-307. GUIHARDataClasses (Class Diagram)

5.33.1.1.1 HAR (Class)

This class is used to represent a Highway Advisory Radio (HAR) device. A HAR is used to broadcast traffic related information over a localized radio transmitter, making the

information available to the traveler. This interface contains methods for getting and setting configuration, getting status, changing communications modes of a HAR, and manipulating and monitoring the HAR in maintenance and online modes.

5.33.1.1.2 HARConfig (Class)

This class holds data pertaining to a HAR device's configuration.

5.33.1.1.3 HARFactory (Class)

This CORBA interface allows new HAR objects to be added to the system. It also allows a requester to acquire a list of HAR objects under the domain of the specific HARFactory object.

5.33.1.1.4 HARStatus (Class)

This class (struct) contains data that indicates the current status of a HAR device. The data contained in this class is that status information which can be transmitted from the HAR to the client as necessary. This struct is also used to within the HAR Service to transmit data to/from the HARControlDB database interface class. (The HAR implementation also contains other private status data elements which are not elements of this class.)

5.33.1.1.5 HISDR1500HAR (Class)

This interface is implemented by objects that provide for the control of an HIS model DR1500 HAR.

5.33.1.1.6 HISDR1500HARConfig (Class)

This class holds configuration data for an HIS model DR1500 HAR.

5.33.1.1.7 ISSAP55HAR (Class)

5.33.1.1.8 ISSAP55HARConfig (Class)

This class holds configuration data for an ISS model AP55 HAR

5.33.1.1.9 SyncHAR (Class)

This class is used to represent a synchronized Highway Advisory Radio (HAR) device. A synchronized HAR can have constituent HARs that it operates in a synchronized mode, allowing a continuous message to be delivered to the motorist as they travel out of range of one HAR and into the range of another.

5.33.1.1.10 SyncHARConfig (Class)

This class holds configuration data for a synchronized HAR.

5.33.1.1.11SynchronizableHAR (Class)

This CORBA interface is implemented by objects that allow for control of HAR devices which can become constituents of a SyncHAR.

5.33.1.1.12SynchronizableHARConfig (Class)

This class holds configuration for a HAR that can operate in a synchronized mode.

5.33.1.1.13WebArbQueue (Class)

This class is a GUI wrapper for a CORBA ArbitrationQueue object.

5.33.1.1.14WebHAR (Class)

This class is a GUI wrapper for a CORBA HAR object.

5.33.1.1.15WebHARConfig (Class)

This class is a wrapper for a HARConfig object.

5.33.1.1.16WebHARFactory (Class)

This class is a wrapper for a HARFactory used to store data pertaining to the HAR factory in the GUI cache.

5.33.1.1.17WebHISDR1500HAR (Class)

This class is a GUI wrapper for a HISDR1500HAR CORBA object.

5.33.1.1.18WebHISDR1500HARConfig (Class)

This class is a wrapper for a HISDR1500HARConfig object.

5.33.1.1.19WebISSAP55HAR (Class)

This class is a GUI wrapper for a ISSAP55HAR CORBA object.

5.33.1.1.20WebISSAP55HARConfig (Class)

This class is a wrapper for an ISSAP55HARConfig object.

5.33.1.1.21WebObjectLocation (Class)

This class provides access to the ObjectLocation struct which contains information about the location of an object in the system.

5.33.1.1.22WebObjectLocationSupporter (Class)

This interface allows common processing for objects supporting an ObjectLocation via the WebObjectLocation wrapper class..

5.33.1.1.23WebSyncHAR (Class)

This class is a GUI wrapper for a SyncHAR CORBA object.

5.33.1.1.24WebSyncHARConfig (Class)

This class is a wrapper for a SyncHARConfig object.

5.33.1.1.25WebSynchronizableHAR (Class)

This class is a GUI wrapper for a SynchronizableHAR CORBA object.

5.33.1.1.26WebSynchronizableHARConfig (Class)

This class is a wrapper for a SynchronizableHARConfig object.

5.34 Chartlite.data.arbqueue-data

5.34.1 Classes

5.34.1.1 chartlite.data.arbqueue_classes (Class Diagram)

This diagram shows classes related to the arbitration queue.

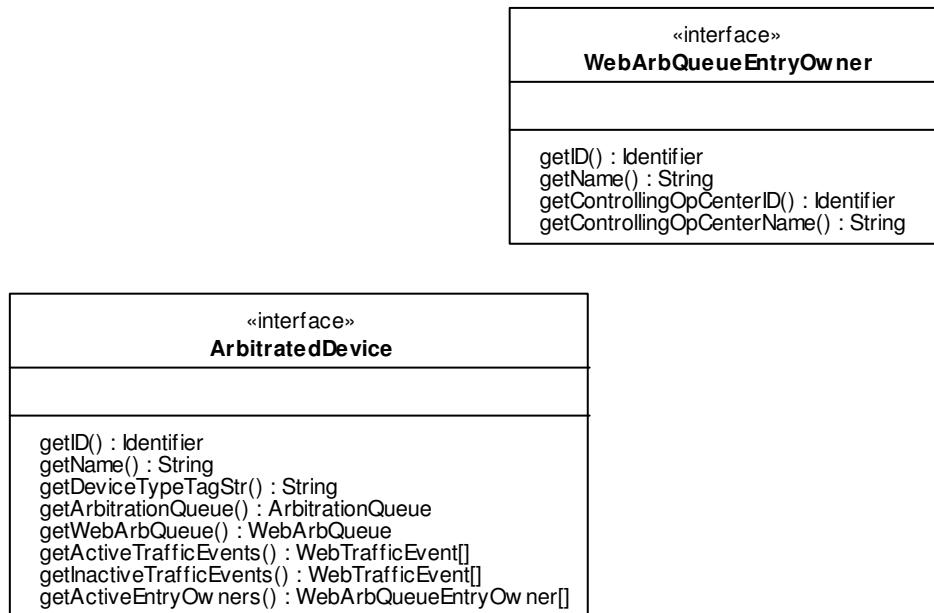


Figure 5-308. chartlite.data.arbqueue_classes (Class Diagram)

5.34.1.1.1 ArbitratedDevice (Class)

This interface allows a class to use a WebArbQueue to track the current state of a device's arbitration queue.

5.34.1.1.2 WebArbQueueEntryOwner (Class)

This interface specifies methods to be implemented by all objects that may place entries on an arbitration queue.

5.35 Chartlite.data.travelroutes-data

5.35.1 Classes

5.35.1.1 GUITravelRouteClasses (Class Diagram)

This diagram shows the classes used by the GUI that are related to travel route management.

5.35.1.1.2 com.vividsolutions.jts.index.quadtree.Quadtree (Class)

This class is a structure that allows data to be indexed geographically.

5.35.1.1.3 CommandQueue (Class)

The CommandQueue class provides a queue for QueueableCommand objects. The CommandQueue has a thread that it uses to process each QueueableCommand in a first in first out order. As each command object is pulled off the queue by the CommandQueue's thread, the command object's execute method is called, at which time the command performs its intended task.

5.35.1.1.4 DataModel (Class)

The data model class serves as a collection of objects. It provides an efficient lookup mechanism for locating any object, and methods which allow for the retrieval of all objects of a particular type. Additionally, this class provides the ability to attach observer objects which are notified when objects are added to or removed from the model. Objects may also notify the DataModel that they have been modified. The model will periodically notify all attached observers of the changes to objects in the model.

5.35.1.1.5 DiscoverTravelRouteClassesCmd (Class)

This class is a queueable command that when executed finds travel route related objects in the CORBA trading service and maintains cached data by either adding new objects to the GUI's data model or by updating the existing cached data.

5.35.1.1.6 DiscoveryDriver (Class)

This class drives the periodic discovery of objects from other services within the CHART system. Other objects in the system that need access to other service's objects add their own QueueableCommand to the DiscoveryDriver. Each time discovery is performed, the discovery driver uses a command queue to execute all queueable commands that have been added in a separate thread of execution. The commands are added to the command queue immediately upon execution, and then executed in serial fashion via the command queue until all commands have executed. The frequency of discovery is controlled by a property. Discovery occurs more frequently immediately after service startup, to more quickly discover objects from other services which may also be starting up at more or less the same time. The DiscoveryDriver can be configured to have multiple threads to allow concurrent discovery of different objects.

5.35.1.1.7 ExtTollSpec (Class)

This structure is used to identify a toll rate route. It contains the supplying external system, the start ID and end ID of the toll rate route (which is the "key" used to identify the toll rate route, and the name by which the external system refers to the route.

5.35.1.1.8 HistoryList<E extends TimeStampedObject> (Class)

This class is used to maintain historical data. It contains limits on the number of entries as well as the maximum entry age. It provides methods to "bucketize" the data to allow data from two or more HistoryLists to be shown using consistent time intervals for the data.

5.35.1.1.9 java.util.Hashtable (Class)

This class implements a hashtable, which is a data structure that maps keys to values. Any non-null object can be used as a key or as a value. Objects used as keys implement the hashCode method which is inherited by all objects from the java.lang.Object class.

5.35.1.1.10 LinkTravTimeHistRecord (Class)

This structure is used to store the most recent X historical link travel time data points acquired by the system. This is a circular array, with the head (oldest record) referenced in the LinkTravelTimeHistStats. The tail (newest record) is head-1 (mod X), and is always the same data point as is contained in the LinkTravelTimeStats structure.

5.35.1.1.11 LinkTravTimeStats (Class)

This structure contains the most recent travel time data point acquired for a roadway link. It matches the most recent record in the LinkTravelTimeHistStats.

5.35.1.1.12 ObjectCache (Class)

The ObjectCache is a wrapper for the DataModel. It provides access to DataModel methods to find objects in the data model, delegating those methods to the DataModel itself. It also provides additional methods of finding name filtered objects and discovering "duplicate" objects (as defined by an isDuplicateOf() method of the Duplicatable interface).

5.35.1.1.13 QueableCommand (Class)

This interface is implemented by objects that can be placed on a command queue.

5.35.1.1.14 RoadwayLinkConfig (Class)

This structure contains the configuration data for a roadway link. It includes the external system name (e.g., "INRIX"), the ID by which the external system identifies the link, and location data.

5.35.1.1.15 RoadwayLinkManager (Class)

This class is used to cache and provide access to roadway link data. This manager class is used instead of just storing roadway link data in the data model because the roadway link data needs to be indexed geographically (not just by ID as supported by the data model). This class is a singleton class and is accessible via the get() method in any code that runs in the same java virtual machine.

5.35.1.1.16RoadwayLinkQuery (Class)

This class holds data that is used to search for links to a travel route. All of the fields are optional, a null value is used to indicate the field is not to be used in the search.

5.35.1.1.17RouteLink (Class)

This structure makes the association between a travel route and one roadway link which helps comprise the route, together with parameters associated with the use of the link within that particular route: the percent of the link to include in the route, and the minimum acceptable quality for link travel time data as used in that particular route.

5.35.1.1.18RouteTollRateHistRecord (Class)

This structure is used to store the most recent X historical route toll rate data points accumulated by the system. This is a circular array, with the head (oldest record) referenced in the RouteTollRateHistStats. The tail (newest record) is head-1 (mod X), and is always the same data point as is contained in the RouteTollRateStats structure.

5.35.1.1.19RouteTollRateStats (Class)

This structure contains the current toll rate data for a travel route. This includes the time the rate became effective and the toll rate itself. This data is also provided in the most recent entry in the history structure. The toll rate field may contain a negative number defined by StatsConstants, which indicates an error. There are two other fields NOT provided in the history structure -- the time the toll rate expires, and a reason string. This will be the empty string if the toll rate has been successfully provided recently, or details on the error condition if an error constant is specified.

5.35.1.1.20RouteTravTimeHistRecord (Class)

This structure is used to store the most recent X historical route travel time data points accumulated by the system. This is a circular array, with the head (oldest record) referenced in the RouteTravelTimeHistStats. The tail (newest record) is head-1 (mod X), and is always the same data point as is contained in the RouteTravelTimeStats structure.

5.35.1.1.21RouteTravTimeStats (Class)

This structure contains the current travel time data for a travel route. This includes the time the travel time was computed, and the computed speed. This data is also provided in the most recent entry in the history structure. The travel time may contain a negative number defined by StatsConstants, which indicates an error. There are two other fields NOT provided in the history structure -- a computed trend (UP, DOWN, or FLAT) and a reason string. This will be the travel time calculation if it has been computed successfully, or details on the error condition if an error constant is specified.

5.35.1.1.22SystemProfileTravelTimeProperties (Class)

This class is a wrapper for SystemProfileProperties that provides easy access to travel time

related properties. It also provides static methods that make it easy to populate a Properties object when setting travel time related properties.

5.35.1.1.23TimeStampedObject (Class)

This interface is implemented by classes that wish to be able to be stored in a HistoryList.

5.35.1.1.24TollRateRouteManager (Class)

This class is used to cache toll rate sources and to provide access to them.

5.35.1.1.25TravelRoute (Class)

This is the primary CORBA interface for working with travel routes in CHART. This interface provides methods for getting various collections of configuration and/or statistical data for a travel route. It also provides methods for objects to register to be TravelRouteConsumer for the travel route (for instances, DMSs that have the route enabled in a traveler information message). Finally it provides methods for updating and removing travel routes.

5.35.1.1.26TravelRouteConfig (Class)

This structure holds the part of the Travel Route configuration pertaining to travel times, if the travel route is configured to track travel times. It contains the IDs of the links comprising the route, but not the link configurations themselves. (See RouteAndLinkConfig.)

5.35.1.1.27TravelRouteFactory (Class)

This interface is the entry point for the Travel Route Management. It serves up travel routes (also an interface) and roadway links (structure data). It provides various operations for acquiring travel routes and roadway links. Since roadway links are not maintained as a separate interface, the factory provides the primary operations for acquiring link configuration and statistical data (although a TravelRoute interface also provides methods for acquiring such data about the links directly associated with it).

5.35.1.1.28TravelRoutePushConsumer (Class)

This class is used to process CORBA events received from a TravelRoute event channel.

5.35.1.1.29TravelTimeRangeDef (Class)

This class holds data for a travel time range definition. It has methods that can convert this object to a JSON object for persistence in the system profile, and to allow its data to be loaded from a JSON object when depersisting from the system profile.

5.35.1.1.30WebCountyInfo (Class)

This class provides access to the CountyInfo struct which contains information about a

county.

5.35.1.1.31WebDMS (Class)

This class represents a dynamic message sign.

5.35.1.1.32WebRoadwayLink (Class)

This class is a wrapper for data pertaining to a Roadway Link.

5.35.1.1.33WebRoadwayLinkConfig (Class)

This class is a wrapper for a RoadwayLinkConfig that allows data from the RoadwayLinkConfig to be accessed from dynamic web pages.

5.35.1.1.34WebRoadwayLinkListItem (Class)

This class is a data holder used to store a WebRoadwayLink and a distance from the end of another link.

5.35.1.1.35WebRoadwayLocation (Class)

This class provides access to the RoadwayLocation struct which contains information about a location on a roadway.

5.35.1.1.36WebStateInfo (Class)

This class provides access to the StateInfo struct which contains information about a state.

5.35.1.1.37WebTollRateRoute (Class)

This class is a wrapper for data from a TollRateRouteInfo object obtained from the travel route factory. It is used to cache data pertaining to a toll rate route in the GUI's data model. (A Toll Rate Route can be used as the toll rate source for a travel route).

5.35.1.1.38WebTollRateStats (Class)

This class is a wrapper for either of the two flavors of toll rate statistics that may be obtained from a travel route. The expiration time is optional and will only be present if constructed from an actual toll rate status (rather than a historical status).

5.35.1.1.39WebTravelRoute (Class)

This class is a wrapper for a CORBA TravelRoute object and is used to cache data pertaining to a travel route in the GUI's data model.

5.35.1.1.40WebTravelRouteConfig (Class)

This class wraps the TravelRouteConfig IDL structure and provides accessor methods.

5.35.1.1.41 WebTravelRouteConsumer (Class)

This interface is implemented by GUI classes that wrap CORBA objects that can act as a TravelRouteConsumer. In R3B3 this will include only WebDMS objects.

5.35.1.1.42 WebTravelRouteFactory (Class)

This class is a wrapper for a CORBA TravelRouteFactory object used to cache data pertaining to the travel route factory in the GUI's data model.

5.35.1.1.43 WebTravelRouteLink (Class)

This class holds data pertaining to a link that is included in a travel route.

5.35.1.1.44 WebTravelRouteStatus (Class)

This class holds status data pertaining to a travel route.

5.35.1.1.45 WebTravelRouteUser (Class)

This interface is implemented by objects that are considered users of travel routes.

5.35.1.1.46 WebTravelTimeStats (Class)

This class is a wrapper for any of the four flavors of travel time data that may be obtained from a travel route. There are two optional fields, trend and quality which are present depending on the structure used to construct the object.

5.35.2 Sequence diagrams

5.35.2.1 chartlite.data.travelroutes.HistoryList:addElement (Sequence Diagram)

This diagram shows the processing that takes place when an element is added to a HistoryList. The new item is added to the tail of a linked list, and then the list is maintained. First, the list is trimmed as needed from the head end (which has the oldest item in the list) to keep the list size within the maximum length. Next, items are trimmed from the head end if their effective time indicates they exceed the maximum age for items in the list.

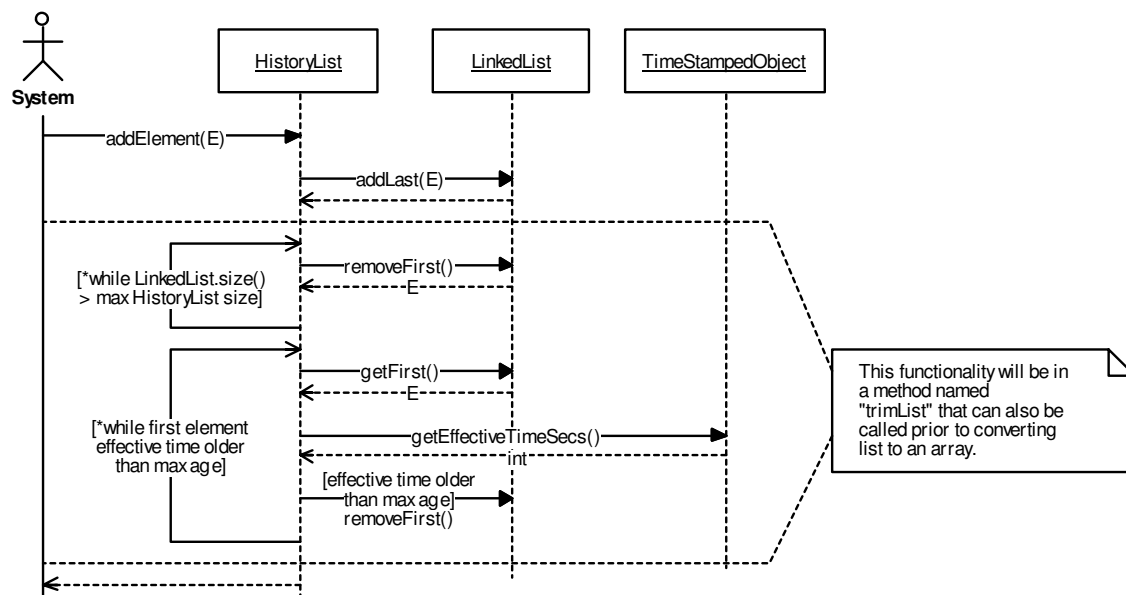


Figure 5-310. chartlite.data.travelroutes.HistoryList:addElement (Sequence Diagram)

5.35.2.2 chartlite.data.travelroutes.HistoryList:toBucketArray (Sequence Diagram)

This diagram shows the processing that is performed to retrieve the entries in a HistoryList in pre-defined time period buckets. An array of "buckets" is passed in to define the time boundary of each bucket in decreasing order (1:00, 12:55, 12:50, etc.). The list of elements in the linked list is first trimmed to make sure we don't have any old elements, and then dumped to an array for processing. The array of elements is processed from oldest to newest because we will store at most 1 element per bucket, so if 2 or more fall into the same bucket the newer element will overwrite the older element. For each element, the list of buckets is checked to find the bucket whose time is greater than the effective time of the element. Once a bucket is found, the element is stored in that bucket, but the bucket index is not moved in case the next element processed will fit in that same bucket. (Because the elements are time ordered, we can be sure that the next element will not belong in a bucket we have already visited). After all elements are processed or the bucket at index zero is not appropriate for an element, the processing stops and the bucketized array is returned. Note that this array can have null elements for buckets that did not have any elements that fit.

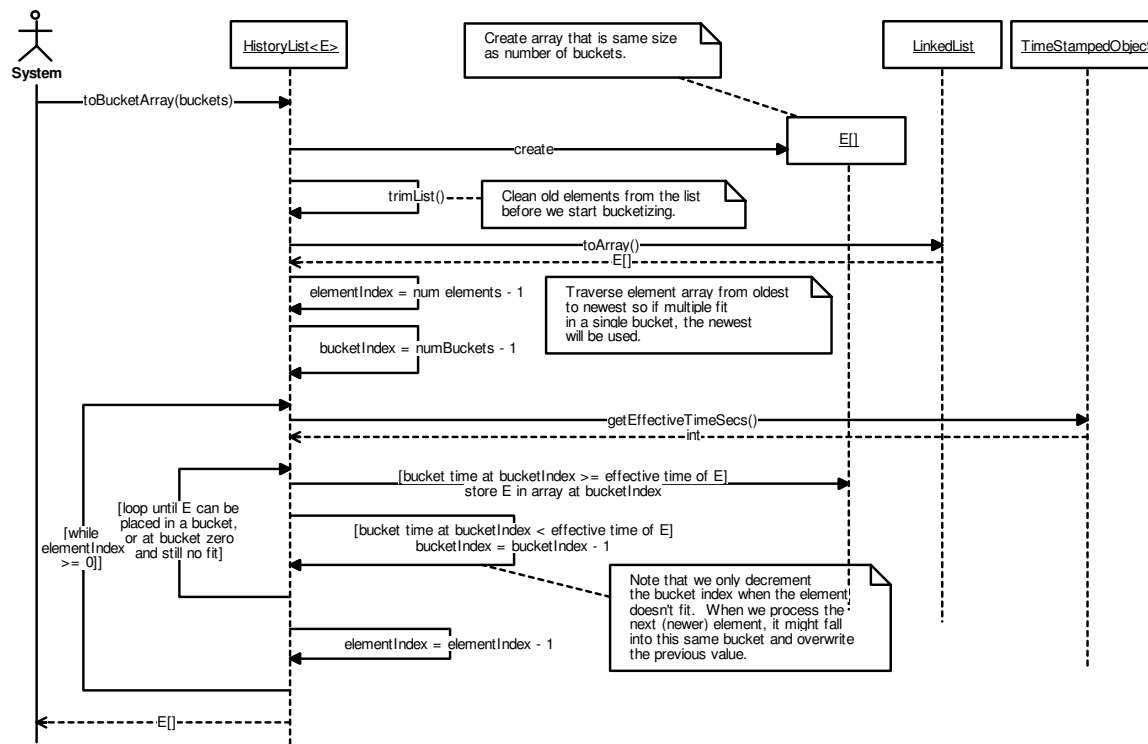


Figure 5-311. chartlite.data.travelroutes.HistoryList:toBucketArray (Sequence Diagram)

5.35.2.3 chartlite.data.travelroutes.RoadwayLinkManager:addOrUpdateLink (Sequence Diagram)

This diagram shows the processing that is performed when the addOrUpdateLink method of the RoadwayLinkManager is called. This method is designed to be called during discovery of roadway links to ensure that new links get added to the manager and existing links get updated as needed. The Hashtable is called to retrieve the WebRoadwayLink object for a given link ID - if the link has been previously added it will exist in the hashtable and be returned, otherwise null will be returned.

If the link already exists in the hashtable, it is called to determine if the current location (start/end points) in the existing WebRoadwayLink match the location in the RoadwayLinkConfigInfo that was passed to this method. If the location data is updated, the WebRoadwayLink is removed from the link starting point and end point quad trees. The data in the WebRoadwayLink is then updated with the data from the RoadwayLinkConfig as needed.

If the link does not already exist in the hashtable, a WebRoadwayLink object is created to wrap the RoadwayLinkConfig and it is stored in the hashtable.

If the link was newly created above, or existed but its location has changed, then the link is added to the starting and ending point quad trees.

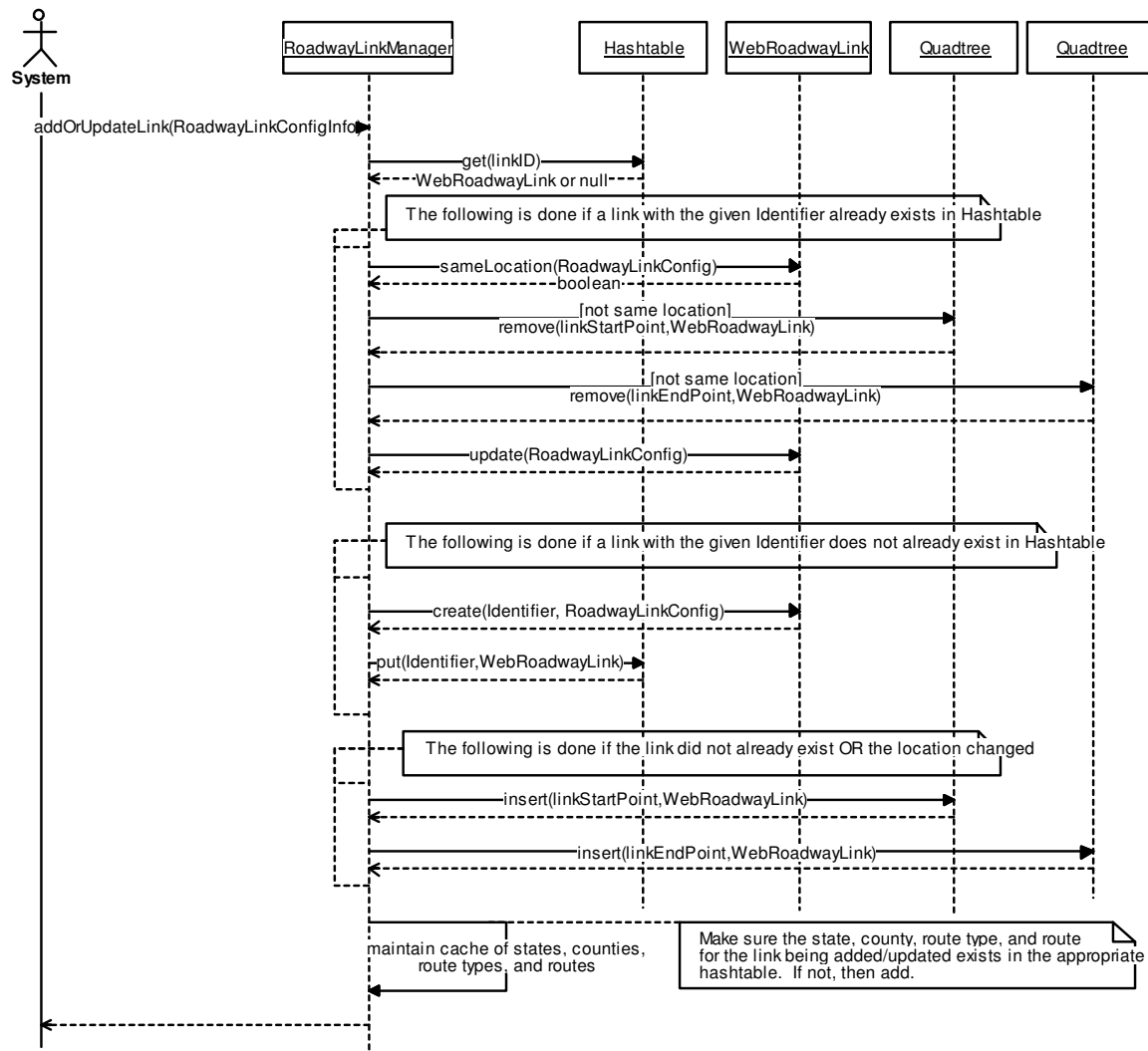


Figure 5-312. chartlite.data.travelroutes.RoadwayLinkManager:addOrUpdateLink (Sequence Diagram)

5.35.2.4 chartlite.data.travelroutes.RoadwayLinkManager:suggestLinks (Sequence Diagram)

This diagram shows the processing that is performed when the suggestLinks method of the RoadwayLinkManager is called. This method is designed to find a path of links that start with the given priorLink. An ArrayList is constructed to hold the links that will be returned, and a query is made into the link starting point quadtree to find the link (or links) that start within the specified threshold distance from the end point of the priorLink.

Each link returned from the query is processed as follows: A check is made to make sure the link is not in the opposing direction of the priorLink, the distance from the prior link is computed, a WebRoadwayLinkListItem is constructed with the WebRoadwayLink from the list and the distance from the prior link, the item is added to a temporary array list. This temporary array list is sorted by distance from the prior link (nearest to furthest) and then each of these links is added to the return link (provided we have not reached the max number of links to return). The priorLink local variable is set to the first WebRoadwayLink in this and will be used for the query performed in the next iteration of the outer loop.

The method ends when the maximum number of suggestions is reached or an iteration results in no new links being added (either none returned from the query, or the ones that were returned were all in the opposing direction).

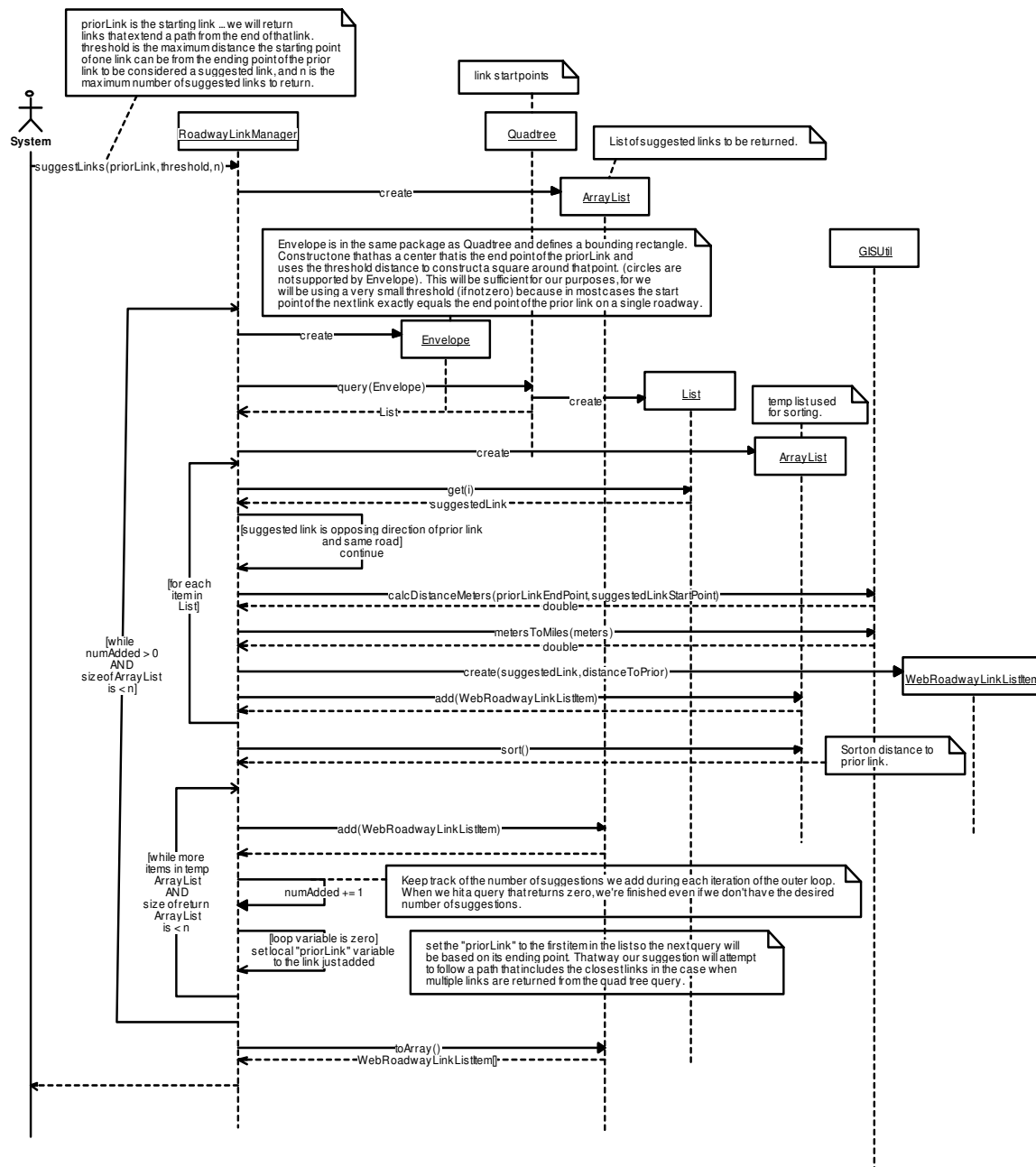


Figure 5-313. chartlite.data.travelroutes.RoadwayLinkManager:suggestLinks (Sequence Diagram)

5.35.2.5 chartlite.data.travelroutes.TravelRoutePushConsumer:routeAdded (Sequence Diagram)

This diagram shows the processing that takes place when the GUI receives a ROUTE_ADDED event. The processing shown takes place after the BasePushConsumer has called the TravelRoutePushConsumer's handleEventData method and that method calls routeAdded to process the event. A new WebTravelRoute object is constructed and its configuration is updated using the configuration data received in the event. The TravelRoute is then called to retrieve the current status data which is then used to update the status in the WebTravelRoute. The new WebTravelRoute object is stored in the data model (GUI cache), and if an object for a travel route with the same ID previously existed the GUI will log a message to warn that an "added" event was received for a travel route that already existed in the cache. Note that this situation is not necessarily a problem, the GUI that was used to add the object to the system could have added it to the cache itself before the corba event was recieved. In any event, we'll want to use the data from the server rather than data that may have been added by the GUI.

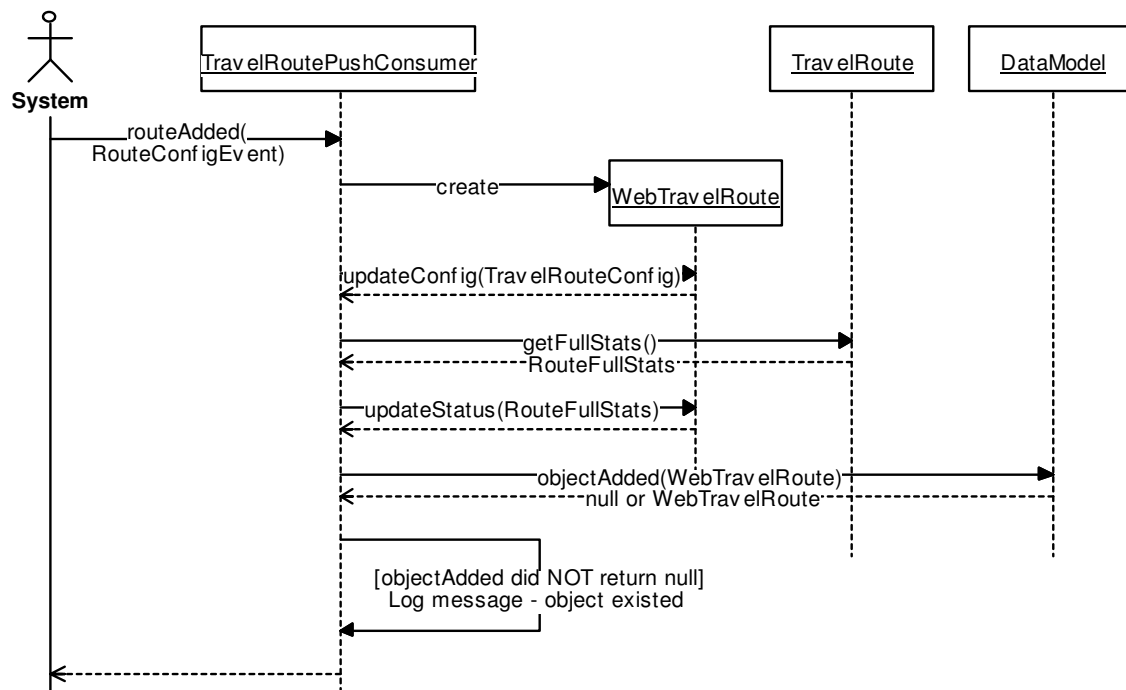
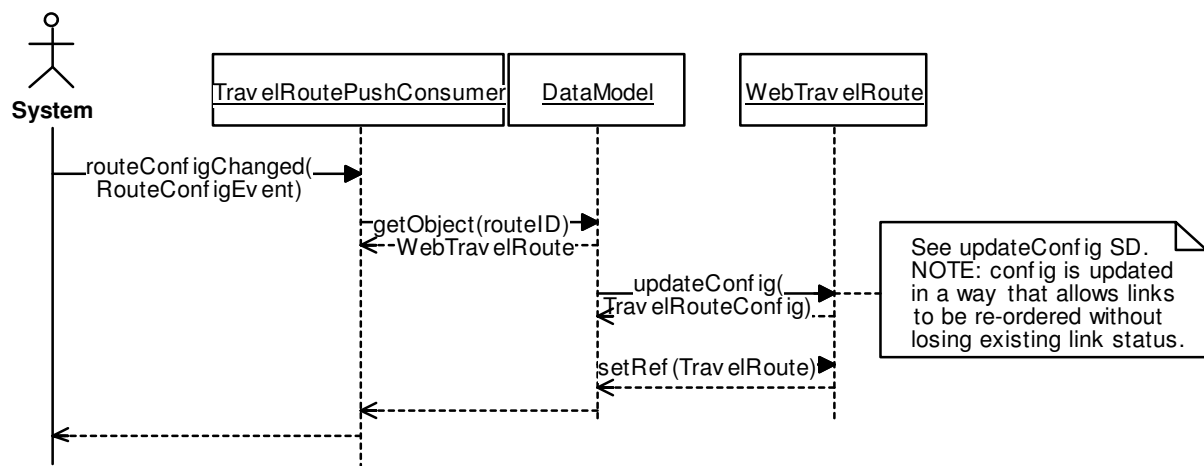


Figure 5-314. chartlite.data.travelroutes.TravelRoutePushConsumer:routeAdded (Sequence Diagram)

5.35.2.6 chartlite.data.travelroutes.TravelRoutePushConsumer:routeConfigChanged (Sequence Diagram)

This diagram shows the processing that takes place when the GUI receives a ROUTE_CONFIG_CHANGED event. The processing shown takes place after the BasePushConsumer has called the TravelRoutePushConsumer's handleEventData method and that method calls routeConfigChanged to process the event. The WebTravelRoute for which the event was pushed is found in the data model, and its updateConfig method is called (see details on updateConfig SD). The TravelRoute CORBA object reference is also stored in the WebTravelRoute, replacing any existing reference.



**Figure 5-315. chartlite.data.travelroutes.TravelRoutePushConsumer:routeConfigChanged
(Sequence Diagram)**

5.35.2.7 chartlite.data.travelroutes.TravelRoutePushConsumer:routeDeleted (Sequence Diagram)

This diagram shows the processing that takes place when the GUI receives a ROUTE_DELETED event. The processing shown takes place after the BasePushConsumer has called the TravelRoutePushConsumer's handleEventData method and that method calls routeDeleted to process the event. The WebTravelRoute object that caches data for the travel route being deleted is retrieved from the data model, and each of its consumers are called to notify them that the route is being deleted. This allows the cached consumer objects to remove their references to the route being deleted. The WebTravelRoute is then removed from the data model.

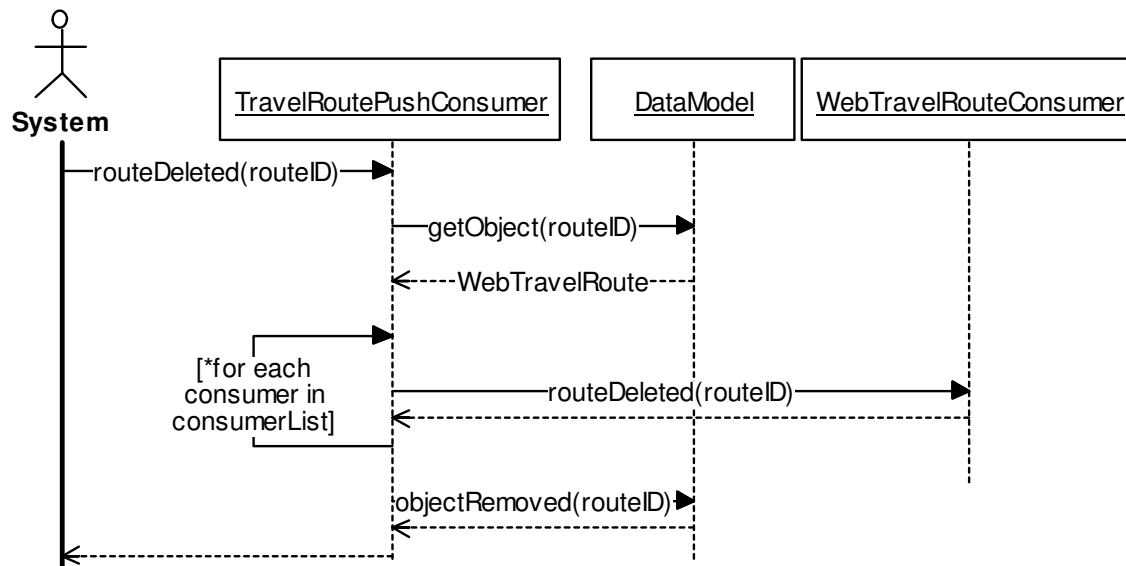


Figure 5-316. chartlite.data.travelroutes.TravelRoutePushConsumer:routeDeleted (Sequence Diagram)

5.35.2.8 chartlite.data.travelroutes.TravelRoutePushConsumer:routeTollRateUpdated (Sequence Diagram)

This diagram shows the processing that takes place when the GUI receives a ROUTE_TOLL_RATE_UPDATED event. The processing shown takes place after the BasePushConsumer has called the TravelRoutePushConsumer's handleEventData method and that method calls routeTollRateUpdated to process the event. The WebTravelRoute for which the event was pushed is found in the data model, and its updateTollRateStats method is called. The WebRouteStatus is called to update the route's toll rate status, which updates the current status and also updates the toll rate history list.

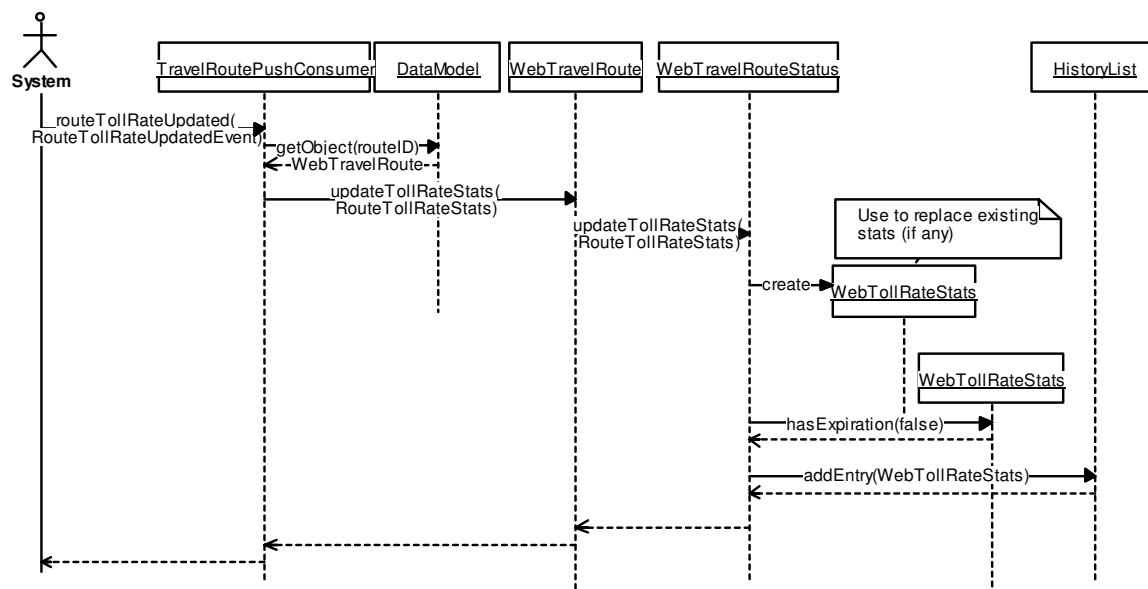


Figure 5-317.
chartlite.data.travelroutes.TravelRoutePushConsumer:routeTollRateUpdated (Sequence Diagram)

5.35.2.9 chartlite.data.travelRoutes.TravelRoutePushConsumer:routeTravelTimeUpdated (Sequence Diagram)

This diagram shows the processing that takes place when the GUI receives a ROUTE_TRAVEL_TIME_UPDATED event. The processing shown takes place after the BasePushConsumer has called the TravelRoutePushConsumer's handleEventData method and that method calls routeTravelTimeUpdated to process the event. The WebTravelRoute for which the event was pushed is found in the data model, and its updateTrvlTimeStats method is called, passing both the route travel time status and the array of travel time statuses for each of the route's links. The WebRouteStatus is called to update the route's travel time status, and it updates the current status and also updates the travel time history list. The WebTravelRoute then processes the travel time status for each link. It retrieves the link from its internal list of links and calls its updateTrvlTimeStats method. That method updates the current travel time stats and also updates the travel time history for the link.

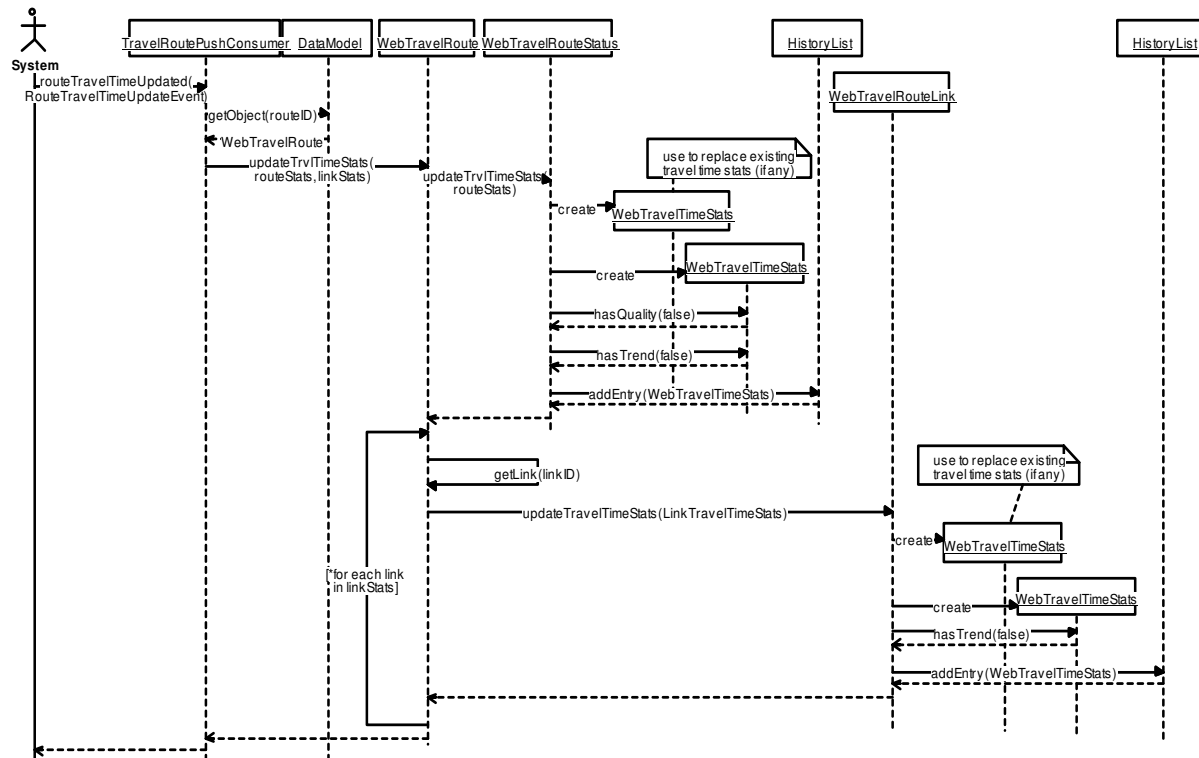
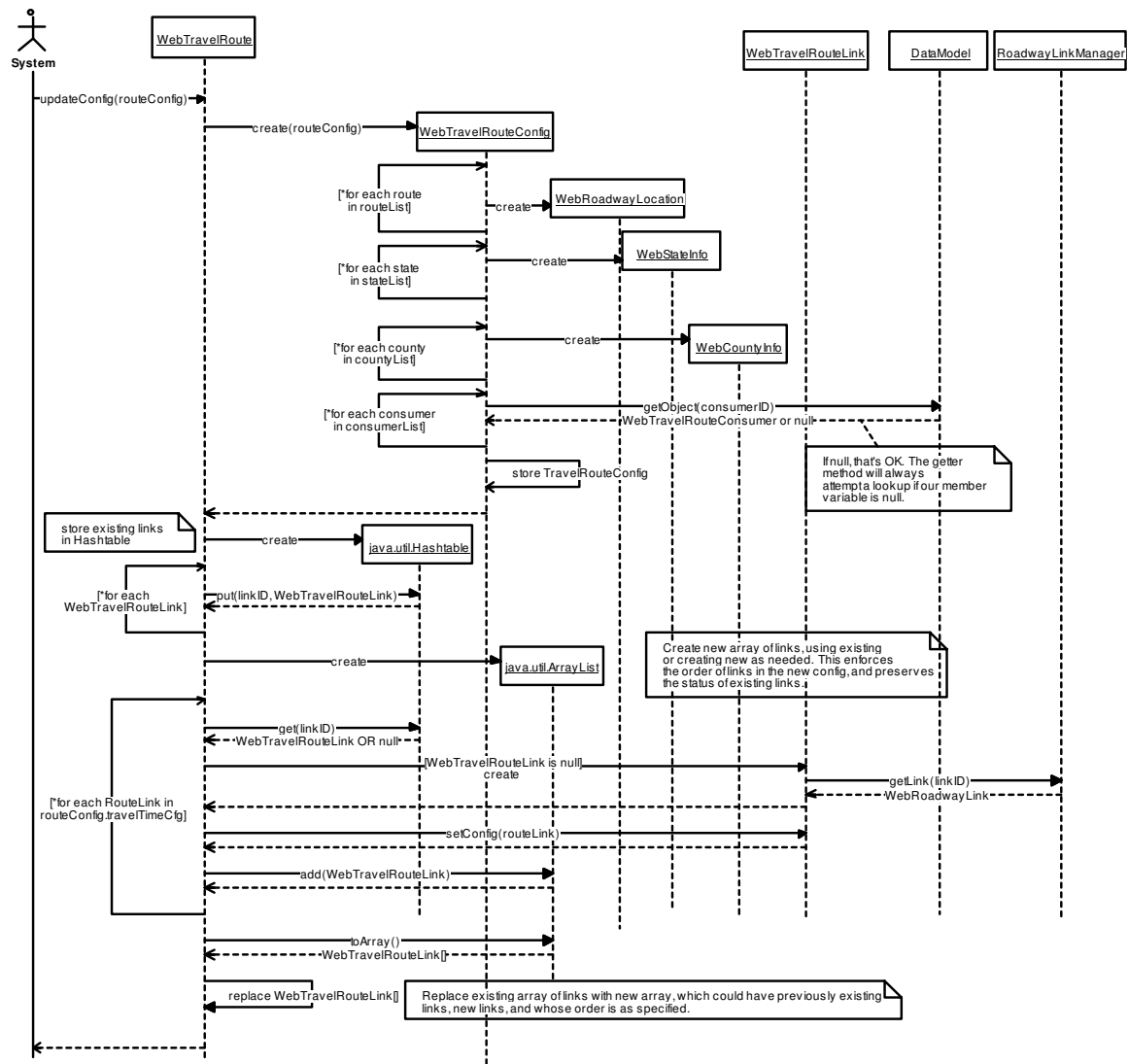


Figure 5-318.
chartlite.data.travelRoutes.TravelRoutePushConsumer:routeTravelTimeUpdated
(Sequence Diagram)

5.35.2.10 `chartlite.data.travelroutes.WebTravelRoute:updateConfig` (Sequence Diagram)

This diagram shows the processing that takes place when the configuration of a `WebTravelRoute` is updated. A new `WebTravelRouteConfig` data is created using the configuration data in the given `TravelRouteConfig` object, creating lists of objects where appropriate. The travel route link configuration data is not processed as part of `WebTravelRouteConfig` construction because the `WebTravelRoute` stores data for each link separate from the `WebTravelRouteConfig`. After the `WebTravelRouteConfig` is created, the `RouteLink` objects in the travel route config's travel time data are processed. First the `WebTravelRoute`'s existing `WebTravelRouteLink` objects are stored in a hashtable for later reference. Each `RouteLink` is then processed. If a `WebTravelRouteLink` object already exists for the link, it is retrieved from the Hashtable and its configuration data is updated. If the link is newly added to the travel route, a new `WebTravelRouteLink` object is created. In either case, the `WebTravelRouteLink` object is stored in an array list. This technique has the effect of reordering the list of links if needed, adding new links, and removing links that no longer exist in the configuration. At the same time, the status data for any existing links is preserved.



5.35.2.11 **chartlite.data.travelroutes.WebTravelRoute:updateStatus (Sequence Diagram)**

This diagram shows the processing that takes place when the WebTravelRoute's updateStatus method is called. This method is provided the full status and is used to completely replace the existing status data for the travel route, including its toll rate status and history, travel time status and history, and the status and history of each of the travel route's links. A new WebTravelRouteStatus object is constructed using data from the RouteFullStats structure that was obtained from the server. It stores the route's toll rate status (if any) in a WebTollRateStats object, and creates a HistoryList<WebTollRateStats> to hold the toll rate history. It then processes each toll rate history record in order (using the oldest record index and looping around to the beginning of the array if needed) and adds each to the HistoryList. Similar processing is done for the route's travel time stats and history. Next, the WebTravelRoutes processes the status for each link. It finds the link in its internal list and then constructs a WebTravelTimeStats object with the new status data and calls the setStatus method of the WebTravelRouteLink. Finally, the WebTravelRoute processes the travel time history for each link. It creates a HistoryList<WebTravelTimeStats> for the link and then processes each history record for the link, adding it to the HistoryList in order. To process the history list in order, we start at the index of the oldest record, proceed to the end of the array, and continue at the beginning if needed. After constructing and populating the new HistoryList, the WebTravelRouteLink's setHistory() method is called to replace any existing history.

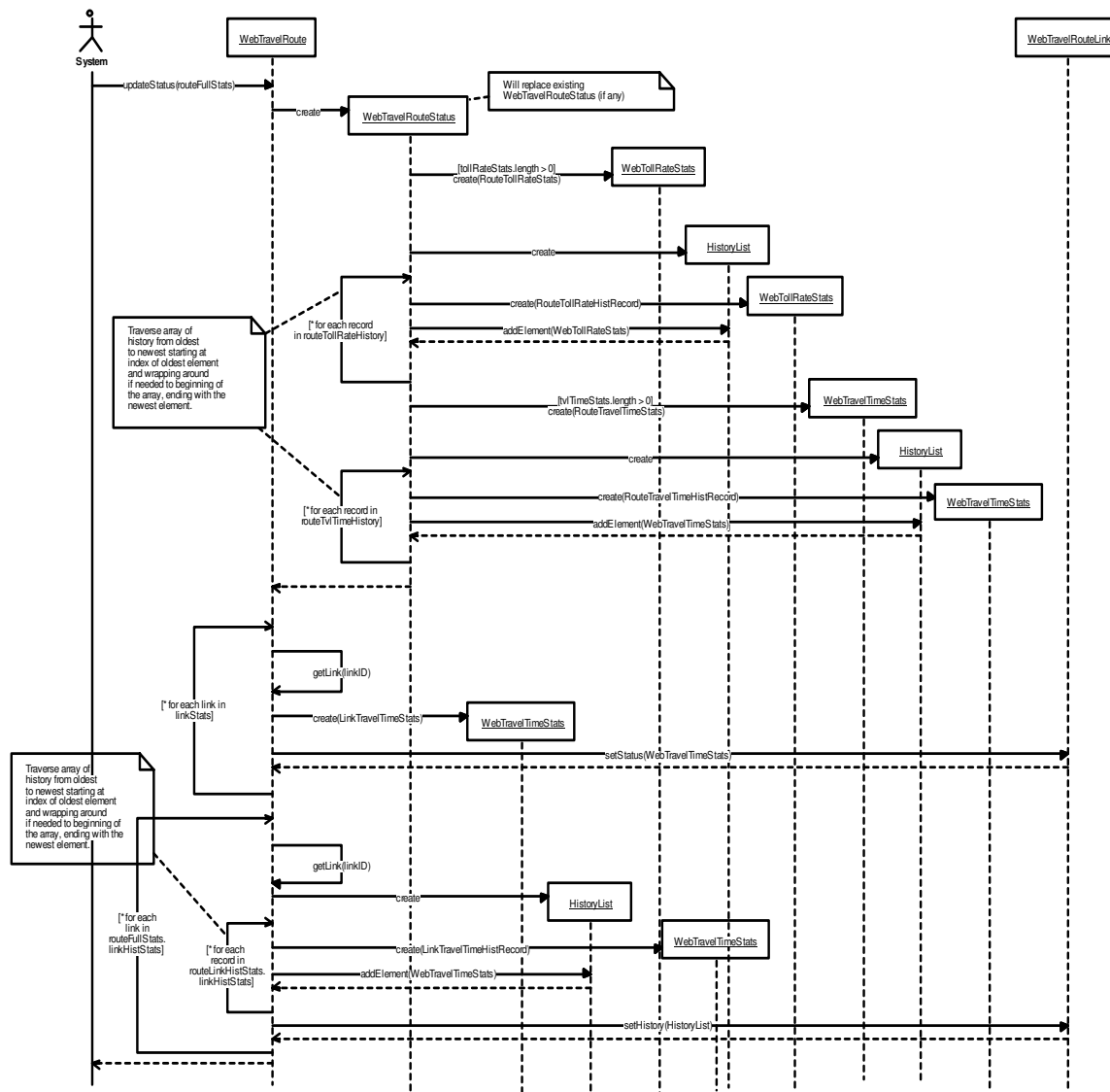


Figure 5-320. chartlite.data.travelroutes.WebTravelRoute:updateStatus (Sequence Diagram)

5.35.2.12 chartlite.data.travelroutes:TravelRouteDiscovery (Sequence Diagram)

This diagram shows the discovery processing related to travel routes. Discovery is the process of finding objects in the CHART server and adding them (or updating them) in the GUI's cache. First travel route event channels are discovered using an existing CHART utility method. Next, travel route factories are discovered. Each factory is called to obtain the lists of roadway links, toll rate routes, and travel routes it serves. Each travel route is called to obtain its configuration and status data. If a travel route does not already exist in the GUI cache, it is added. Otherwise the existing cached object is updated with the new configuration and status data.

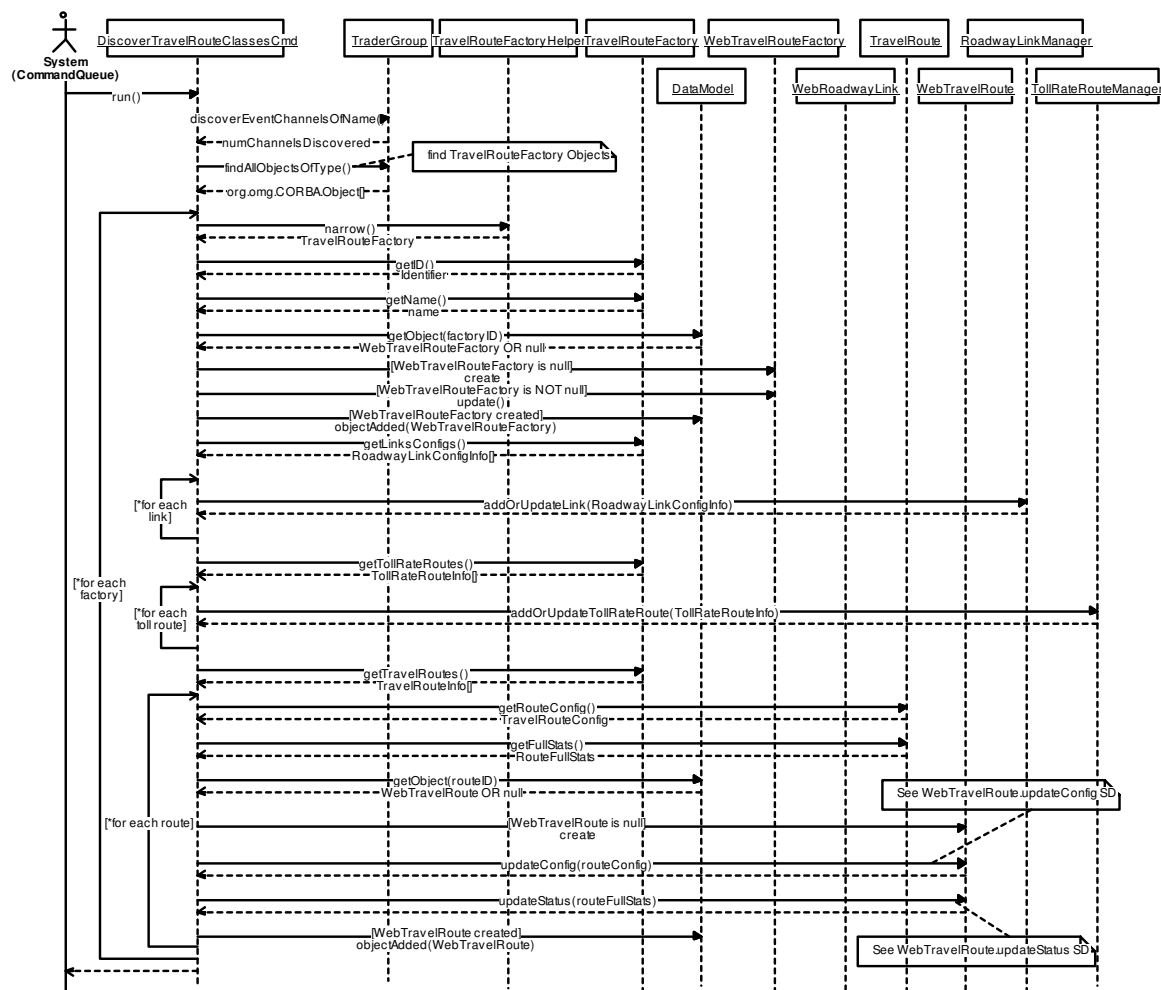


Figure 5-321. chartlite.data.travelroutes:TravelRouteDiscovery (Sequence Diagram)

5.36 Chartlite.data.templates-data

5.36.1 Classes

5.36.1.1 GUIMessageTemplateDataClasses (Class Diagram)

This diagram shows GUI classes related to traveler information message template data.

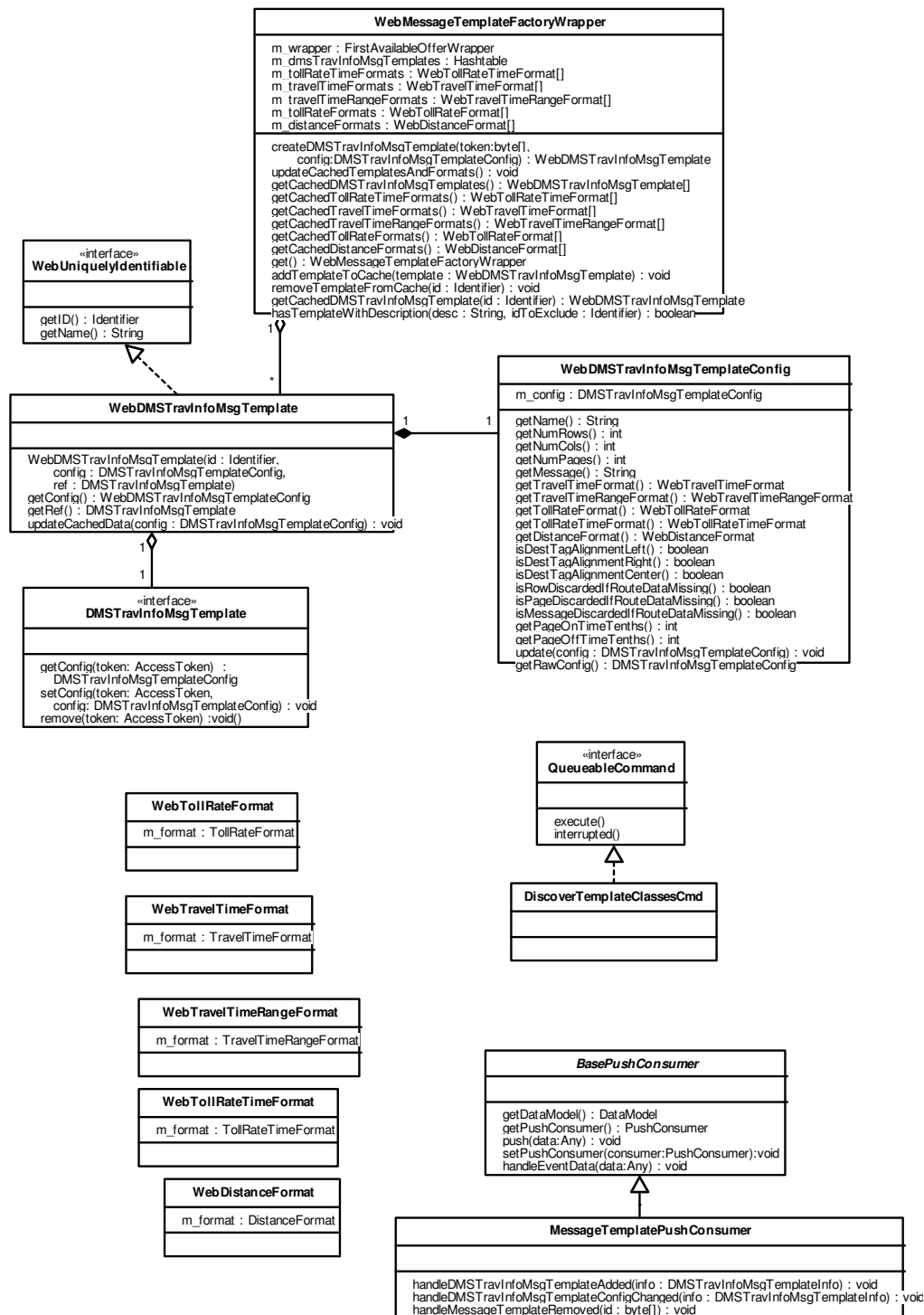


Figure 5-322. GUIMessageTemplateDataClasses (Class Diagram)

5.36.1.1.1 BasePushConsumer (Class)

This is a base class for push consumers. Derived classes must implement

handleEventData().

5.36.1.1.2 DiscoverTemplateClassesCmd (Class)

This class is called to periodically discover message template classes from the trading service and CHART Message Utility Service.

5.36.1.1.3 DMSTravInfoMsgTemplate (Class)

The DMSTravInfoMsgTemplate interface is implemented by objects that allow execution of tasks associated with DMS travel information message templates.

5.36.1.1.4 MessageTemplatePushConsumer (Class)

5.36.1.1.5 QueueableCommand (Class)

A QueueableCommand is an interface used to represent a command that can be placed on a CommandQueue for asynchronous execution. Derived classes implement the execute method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a CommandQueue. The CommandQueue driver calls the execute method to execute a command in the queue and a call to the interrupted method is made when a CommandQueue is shut down.

5.36.1.1.6 WebDistanceFormat (Class)

5.36.1.1.7 WebDMSTravInfoMsgTemplate (Class)

This class wraps the DMSTravInfoMsgTemplate CORBA object representing a message template. It caches the data represented by the remote object and provides accessors for easy access to the cached data.

5.36.1.1.8 WebDMSTravInfoMsgTemplateConfig (Class)

This class wraps the DMSTravInfoMsgTemplateConfig CORBA structure and provides accessor methods for use in the GUI.

5.36.1.1.9 WebMessageTemplateFactoryWrapper (Class)

This class provides access to functionality provided by the MessageTemplateFactory CORBA objects in the system. It is also used to store cached data from the factories including the data formats and message templates in the system.

5.36.1.1.10WebTollRateFormat (Class)

5.36.1.1.11WebTollRateTimeFormat (Class)

5.36.1.1.12WebTravelTimeFormat (Class)

5.36.1.1.13WebTravelTimeRangeFormat (Class)

5.36.1.1.14WebUniquelyIdentifiable (Class)

This interface provides functionality for GUI objects that represent UniquelyIdentifiable objects as defined in the IDL.

5.36.2 Sequence Diagrams

5.36.2.1 `chartlite.data.templatemanagement:discoverTemplateClasses` (Sequence Diagram)

This diagram shows how DMS traveler information message templates and the available data formats are discovered. The discovery driver will periodically call the `DiscoverTemplateClassesCmd` to execute. The event channels are queried from the CORBA Trading Service, and they are added to the event consumer group to register the consumers and monitor the event channels. The `GUIMessageTemplateFactoryWrapper` is called to update the cached templates and formats. It uses a `FirstAvailableOfferWrapper` to call the first available `MessageTemplateFactory` (or call one that is known to work, if the first one recently failed). The factory is queried to obtain the templates, which are replicated in the database. If a template is already cached but is not in the newly obtained list, it will be removed from the cache. If the template is already cached, it will be updated. If it is not in the cache, a new `WebDMSTravInfoMsgTemplate` object is created and added to the cache. The factory is then called to get each type of format. The formats are wrapped in GUI-specific wrapper objects to allow easier access, and these are cached for future use.

5.36.2.2 chartlite.data.templatemanagement:handleEventData (Sequence Diagram)

This diagram shows the processing when a message template CORBA event is received. If the event type indicates that a new message template was added, a new WebDMSTravInfoMsgTemplate object is created and added to the cache in the WebMessageTemplateFactoryWrapper. If the event type indicates that the template configuration was changed, the template wrapper is retrieved from the cache and called to update its configuration in memory. If the event type indicates that a template was removed, the template wrapper object is removed from the cache.

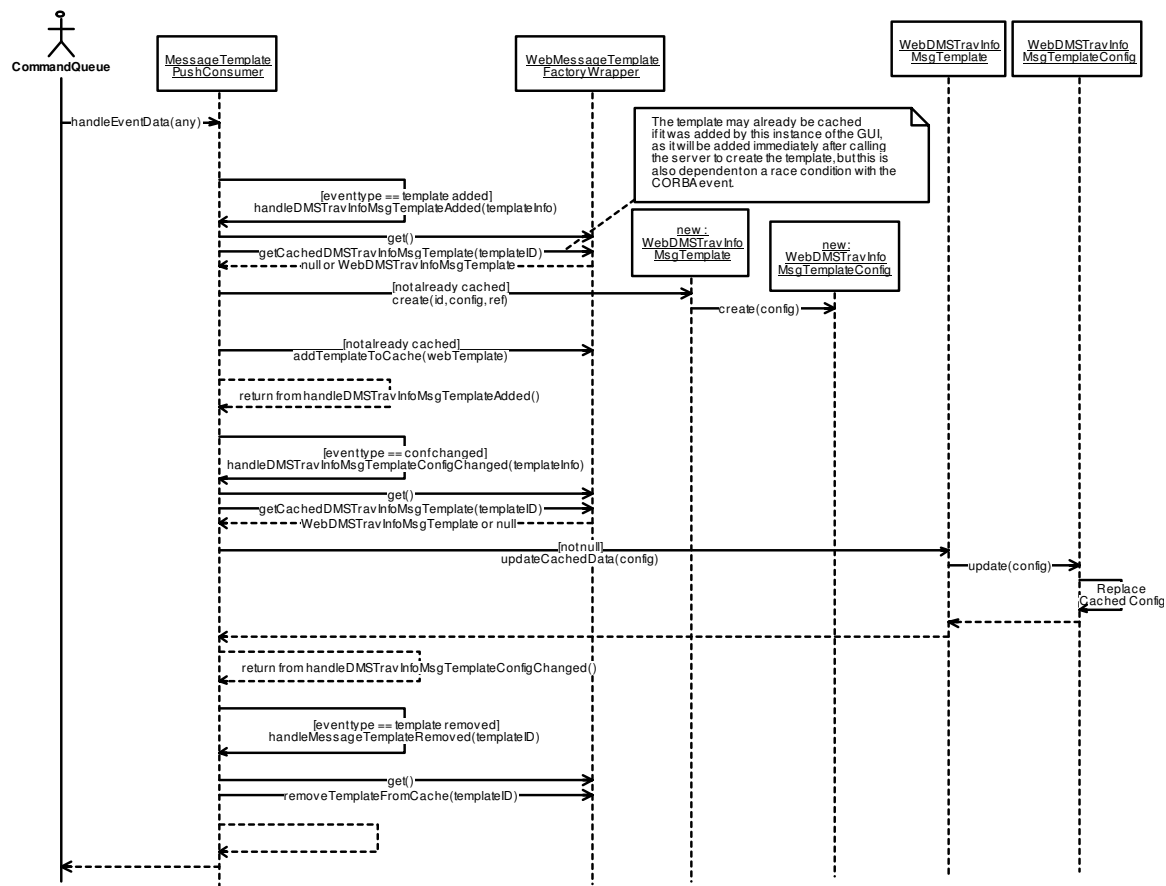


Figure 5-324. chartlite.data.templatemanagement:handleEventData (Sequence Diagram)

5.37 Chartlite.data.geoareas-data

5.37.1 Classes

5.37.1.1 GUIGeoAreaClasses (Class Diagram)

This diagram contains classes related to geographic areas.

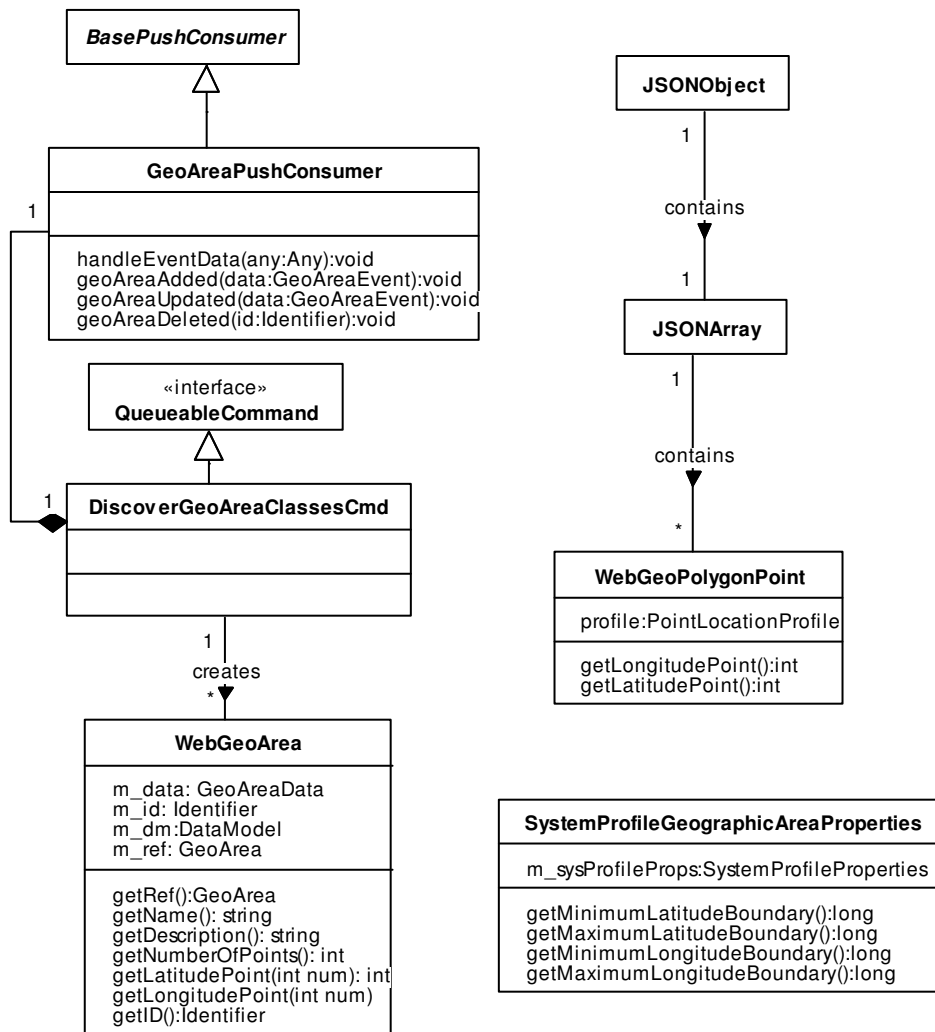


Figure 5-325. GUIGeoAreaClasses (Class Diagram)

5.37.1.1.1 BasePushConsumer (Class)

This is a base class for push consumers. Derived classes must implement

handleEventData().

5.37.1.1.2 DiscoverGeoAreaClassesCmd (Class)

This diagram contains data classes related to geographic areas.

5.37.1.1.3 GeoAreaPushConsumer (Class)

This push consumer handles events related to adding, updating, or removing geographic areas.

5.37.1.1.4 JSONArray (Class)

An array of JSONObject objects.

5.37.1.1.5 JSONObject (Class)

This class is a utility that provides methods to allow for creating and parsing strings that use JSON notation to represent an object's data elements.

5.37.1.1.6 QueueableCommand (Class)

A QueueableCommand is an interface used to represent a command that can be placed on a CommandQueue for asynchronous execution. Derived classes implement the execute method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a CommandQueue. The CommandQueue driver calls the execute method to execute a command in the queue and a call to the interrupted method is made when a CommandQueue is shut down.

5.37.1.1.7 SystemProfileGeographicAreaProperties (Class)

This class contains system profile properties related to geographic areas.

5.37.1.1.8 WebGeoArea (Class)

This class wraps a GeoArea object to provide methods for accessing the data from a velocity template.

5.37.1.1.9 WebGeoPolygonPoint (Class)

This helper class wraps a PointLocationProfile structure.

5.37.2 Sequence Diagrams

5.37.2.1 chartlite.data.geoareamgmt:discoverGeoAreaClasses (Sequence Diagram)

This diagram shows how geographic areas are discovered. The discovery driver will periodically call the DiscoverGeoAreasCmd to execute. The event channels are queried from the CORBA Trading Service, and they are added to the event consumer group to register the consumers and monitor the event channels. The factory is queried to obtain the geo areas, which are replicated in the database. If a geo area is already cached but is not in the newly obtained list, it will be removed from the cache. If the geo area is already cached, it will be updated. If it is not in the cache, a new WebGeoArea object is created and added to the cache.

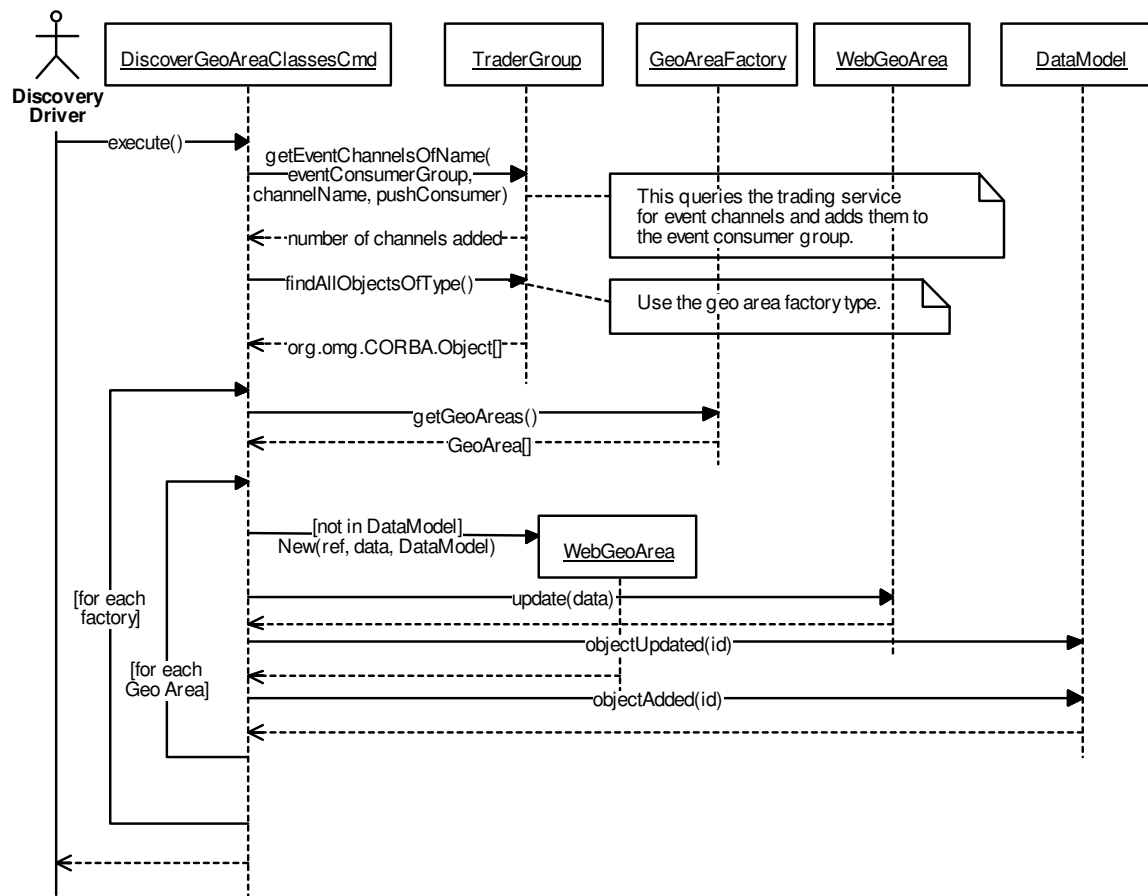


Figure 5-326. chartlite.data.geoareamgmt:discoverGeoAreaClasses (Sequence Diagram)

5.38 Chartlite.data.alerts-data

5.38.1 Classes

5.38.1.1 data.alerts.classes (Class Diagram)

This diagram shows classes related to alerts that are used to store alerts in the data model. For R3B2, one new alert type is being added, as annotated on the diagram. The remainder of the classes shown on this diagram existed prior to R3B2.

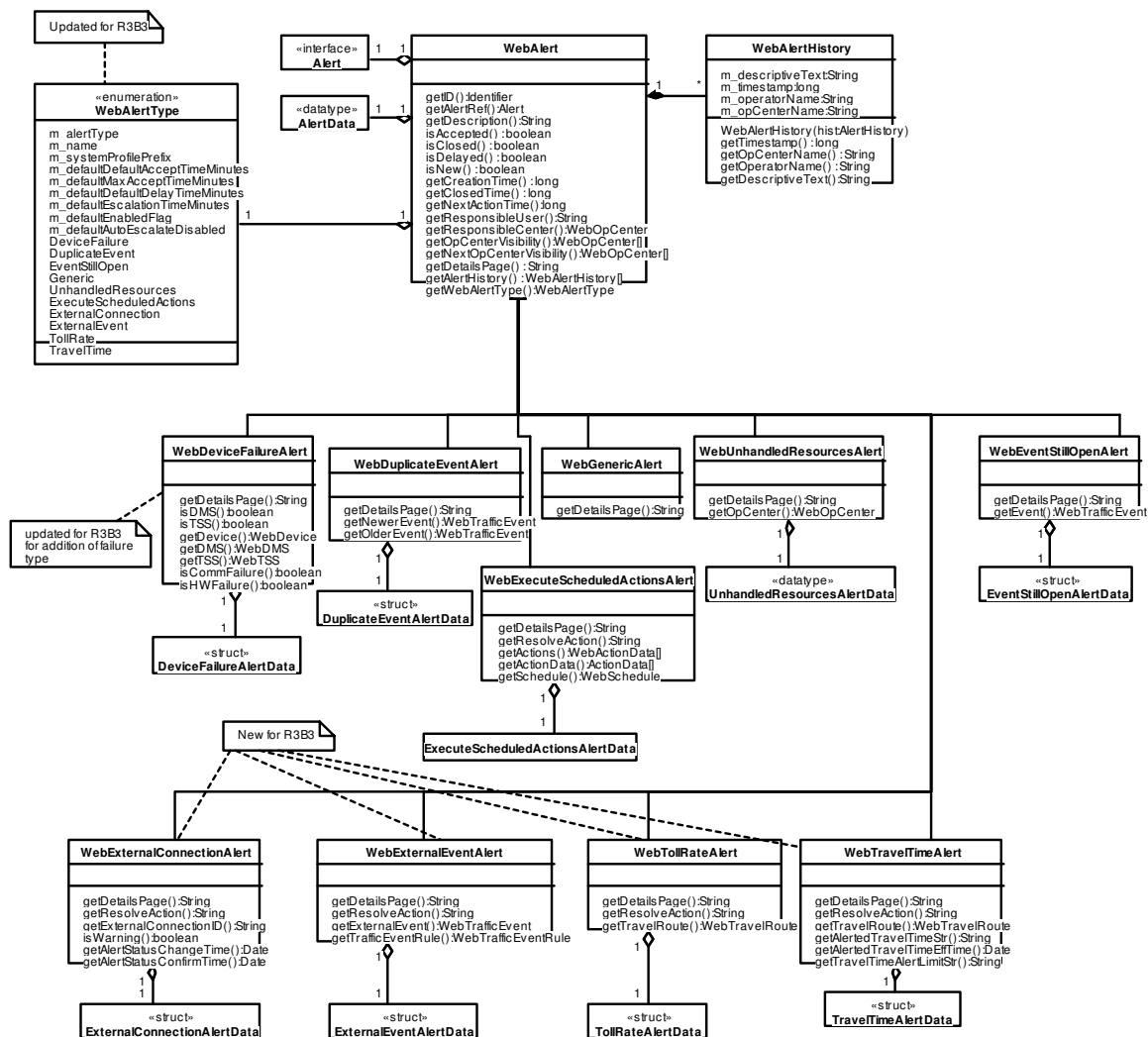


Figure 5-327. data.alerts.classes (Class Diagram)

5.38.1.1.1 Alert (Class)

This is a CORBA interface that provides access to information pertaining to an Alert and provides operations used to manage an alert.

5.38.1.1.2 AlertData (Class)

This is a CORBA struct, defined in IDL, that contains the data that applies to all alert types.

5.38.1.1.3 DeviceFailureAlertData (Class)

This is a CORBA struct, defined in IDL, that contains base alert data plus data specific to a DeviceFailureAlert. Specific to this alert is the traffic event id of the failed device event causing the alert. Also included is information on the device failure type.

5.38.1.1.4 DuplicateEventAlertData (Class)

This is a CORBA struct, defined in IDL, that contains base alert data plus data specific to a DuplicateEventAlert. Specific to this alert are the event ids of the two probable duplicate traffic events.

5.38.1.1.5 EventStillOpenAlertData (Class)

This is a CORBA struct, defined in IDL, that contain the base alert data plus data specific to an EventStillOpenAlert. Specific to this alert is the id of the traffic event that is still open.

5.38.1.1.6 ExecuteScheduledActionsAlertData (Class)

This is a CORBA struct, defined in IDL, that contains the base alert data plus data specific to an ExecuteScheduledActionsAlert.

5.38.1.1.7 ExternalConnectionAlertData (Class)

This IDL structure contains data specific to an External Connection Alert, e.g., the ID of the interface which is having trouble and a flag indicating whether the connection is in failure or warning status, the timestamp it transitioned. (The GUI displays additional data which is best acquired from the GUI's object cache.) (Text in the base AlertData structure provides a textual description and alert management data.)

5.38.1.1.8 ExternalEventAlertData (Class)

This IDL structure contains data specific to an External Event Alert, e.g., the ID of the event and the ID of the first rule found that requested an alert be sent. (Text in the base AlertData structure provides a textual description and alert management data.)

5.38.1.1.9 TollRateAlertData (Class)

This IDL structure contain data specific to a Toll Rate Alert, e.g., the travel route which no longer has data for its toll rate. (Text in the base AlertData structure provides a textual description and alert management data.)

5.38.1.1.10TravelTimeAlertData (Class)

This IDL structure contains data specific to a Travel Time Alert, e.g., the travel time limit and the travel time which exceeded the limit. (Text in the base AlertData structure provides a textual description and alert management data.)

5.38.1.1.11UnhandledResourcesAlertData (Class)

This is a CORBA struct, defined in IDL, that contains the base alert data plus data specific to an UnhandledResourcesAlert.

5.38.1.1.12WebAlert (Class)

This class is used to wrap a CORBA Alert object so that its data may be cached in the CHART GUI servlet and to allow its data to be accessed from within a Velocity template.

5.38.1.1.13WebAlertHistory (Class)

This class is used to wrap AlertHistory data to allow it to be accessed from within a Velocity template.

5.38.1.1.14WebAlertType (Class)

This enumeration identifies the alert types supported by the system along with information specific to each alert type that helps in using generic code to process all alert types. For R3B2 the ExecuteScheduledActions alert type is added.

5.38.1.1.15WebDeviceFailureAlert (Class)

This class is used to wrap a DeviceFailureAlert CORBA object and provide access to data that is specific to this type of alert.

5.38.1.1.16WebDuplicateEventAlert (Class)

This class is used to wrap a DuplicatEventAlert and provide access to its type specific data.

5.38.1.1.17WebEventStillOpenAlert (Class)

This class is used to wrap an EventStillOpenAlert and provide access to its type specific data.

5.38.1.1.18WebExecuteScheduledActionsAlert (Class)

This class is used to cache data for an ExecuteScheduledActionsAlert in the GUI. It provides access to the alert data and overrides the abstract methods of WebAlert to provide a details page and resolve action specific to this alert type.

5.38.1.1.19WebExternalConnectionAlert (Class)

This class is a GUI wrapper for an ExternalConnection alert. It provides access to data

contained in an ExternalConnectionAlertData object.

5.38.1.1.20WebExternalEventAlert (Class)

This class is a GUI wrapper for an ExternalEventAlert. It provides access to data contained in an ExternalEventAlertData object.

5.38.1.1.21WebGenericAlert (Class)

This class is used to wrap a GenericAlert (manual alert).

5.38.1.1.22WebTollRateAlert (Class)

This class is a GUI wrapper for a TollRateAlert. It provides access to data contained in a TollRateAlertData object.

5.38.1.1.23WebTravelTimeAlert (Class)

This class is a GUI wrapper for a TravelTimeAlert. It provides access to data contained in an TravelTimeAlertData object.

5.38.1.1.24WebUnhandledResourcesAlert (Class)

This class is used to wrap an UnhandledResourcesAlert and provide access to its type specific data.

5.39 Chartlite.data.externalsystem-data

5.39.1 Classess

5.39.1.1 GUIExternalSystemClasses (Class Diagram)

This diagram contains data classes related to external interfaces.

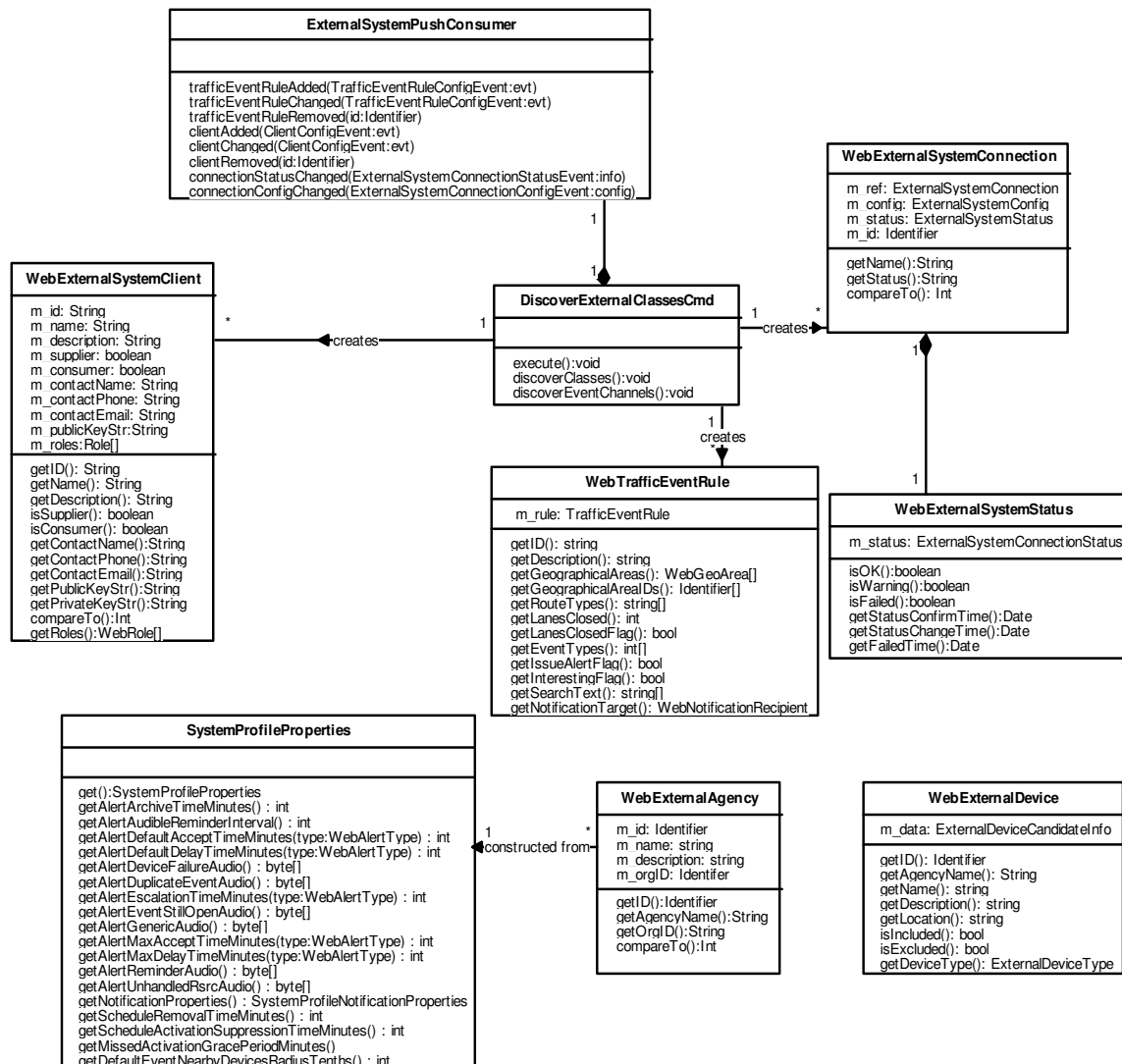


Figure 5-328. GUIExternalSystemClasses (Class Diagram)

5.39.1.1.1 DiscoverExternalClassesCmd (Class)

This class is called to periodically discover classes related to the external system from the trading service and External Interface module. This includes traffic event rules, and external connections

5.39.1.1.2 ExternalSystemPushConsumer (Class)

This event consumer class handles events related to traffic event rules, external system clients, and external connections

5.39.1.1.3 SystemProfileProperties (Class)

This class is used to cache the system profile properties and provide access to them. It is also used to interact with the server to change system profile settings.

5.39.1.1.4 WebExternalAgency (Class)

This class wraps an External Agency for display in the GUI.

5.39.1.1.5 WebExternalDevice (Class)

This class wraps an External Device for display in the GUI.

5.39.1.1.6 WebExternalSystemClient (Class)

This class wraps an external system client for display in the GUI

5.39.1.1.7 WebExternalSystemConnection (Class)

This class wraps an external system connection for display in the GUI.

5.39.1.1.8 WebExternalSystemStatus (Class)

This class wraps an ExternalSystemConnectionStatus class for display in the GUI

5.39.1.1.9 WebTrafficEventRule (Class)

This class wraps a TrafficEventRule for display in the GUI

5.39.2 Sequence Diagrams

5.39.2.1 chartlite.data.externalsystem:DiscoverExternalSystemClasses (Sequence Diagram)

This diagram shows the processing that occurs for initial and periodic discovery of external system classes. The external system classes consist of traffic event inclusion rules and external connections.

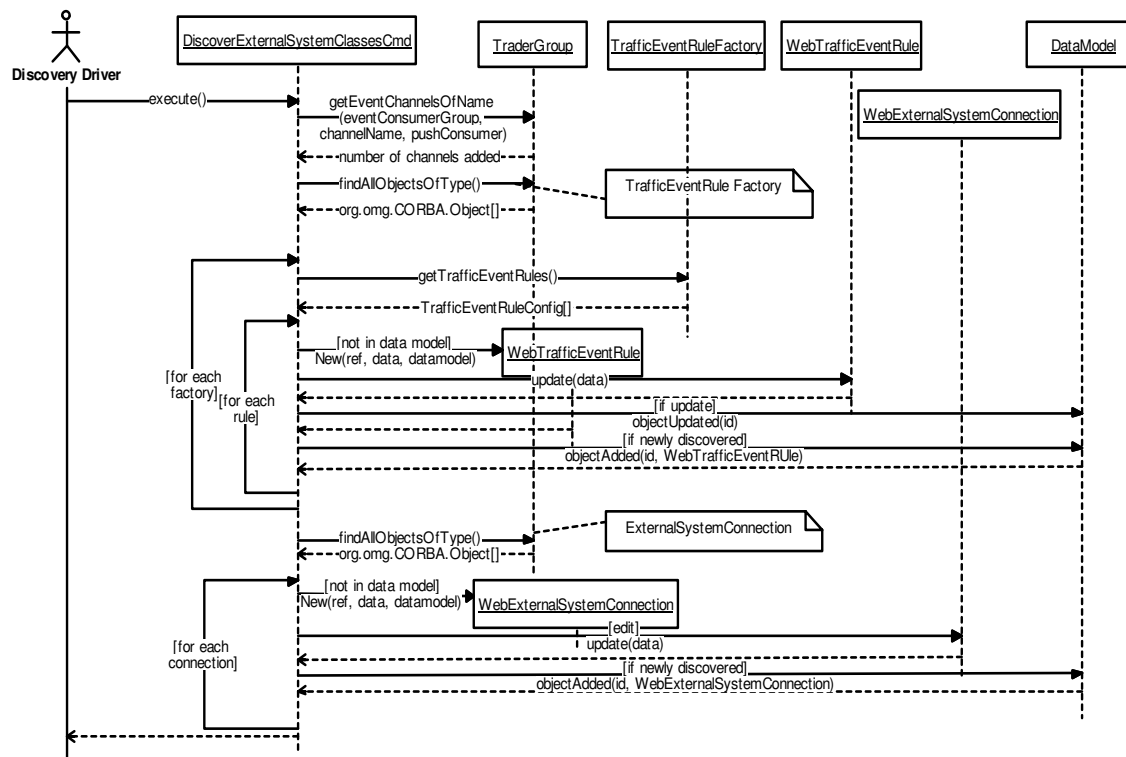


Figure 5-329. chartlite.data.externalsystem:DiscoverExternalSystemClasses (Sequence Diagram)

5.39.2.2 ExternalSystemPushConsumer:clientAdded (Sequence Diagram)

This diagram shows the processing that occurs when a event comes in that indicates a new external client was added.

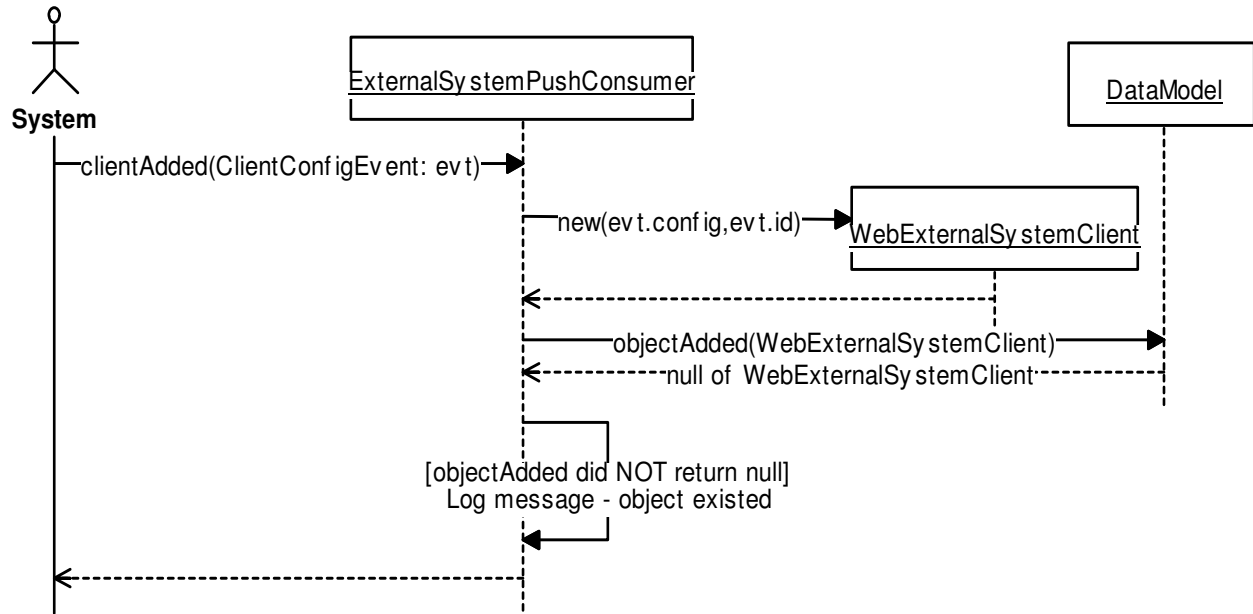


Figure 5-330. ExternalSystemPushConsumer:clientAdded (Sequence Diagram)

5.39.2.3 ExternalSystemPushConsumer:clientRemoved (Sequence Diagram)

This diagram shows the processing that occurs when a client removed event is received on the event channel.

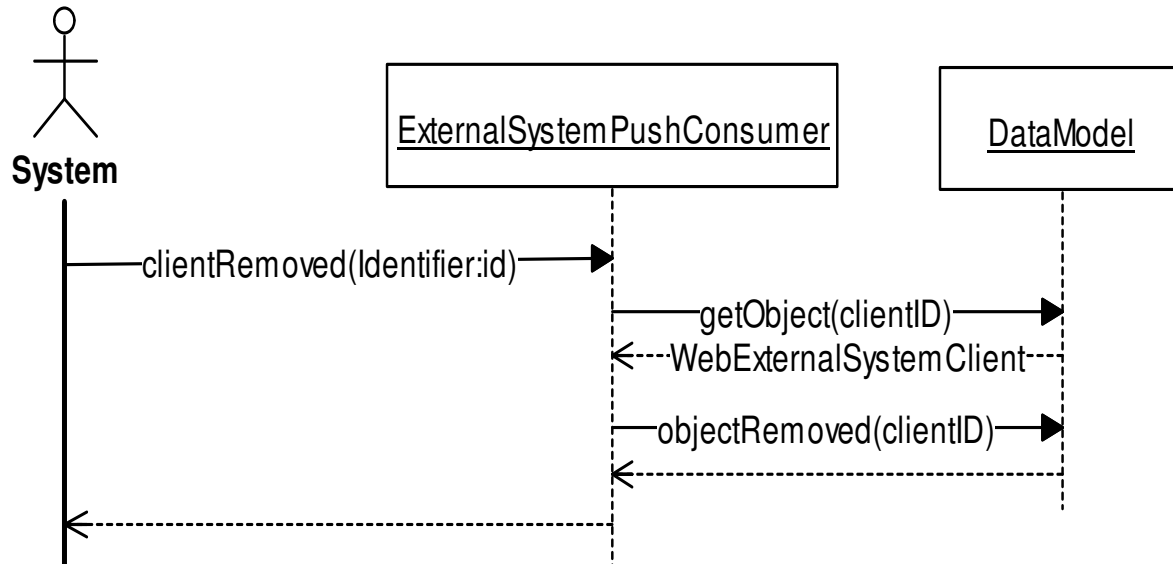


Figure 5-331. ExternalSystemPushConsumer:clientRemoved (Sequence Diagram)

5.39.2.4 ExternalSystemPushConsumer:clientUpdated (Sequence Diagram)

This diagram shows the processing that occurs when a client updated event comes in on the event channel.

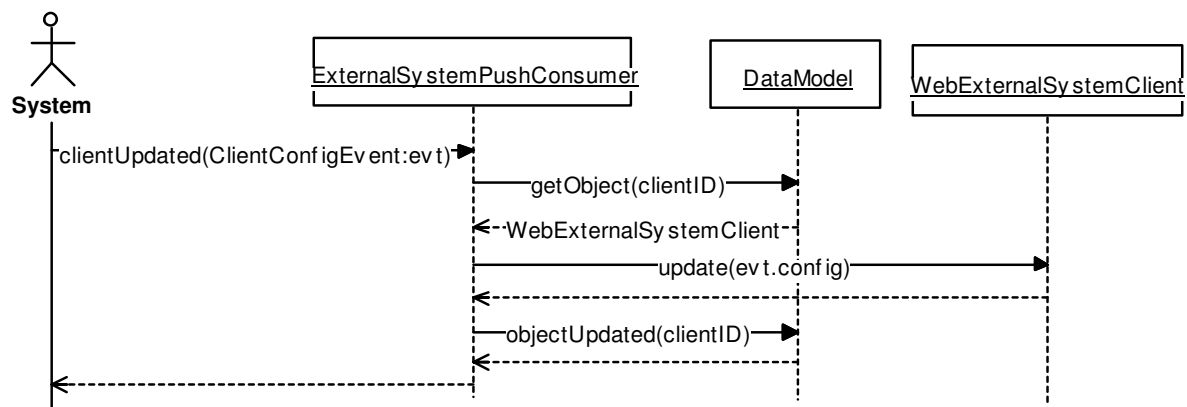


Figure 5-332. ExternalSystemPushConsumer:clientUpdated (Sequence Diagram)

5.39.2.5 ExternalSystemPushConsumer:connectionStatusChanged (Sequence Diagram)

This diagram shows the processing that occurs when a connection status changed event comes in.

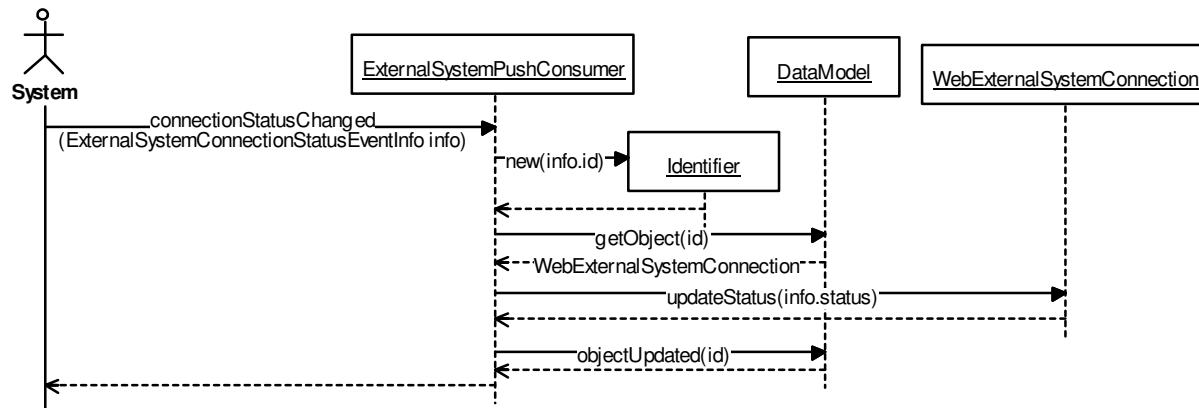


Figure 5-333. ExternalSystemPushConsumer:connectionStatusChanged (Sequence Diagram)

5.40 Chartlite.data.tss-data

5.40.1 Classes

5.40.1.1 GUITSSDataClasses (Class Diagram)

This diagram shows objects related to adding TCP/IP connection functionality and geo location to TSS's.

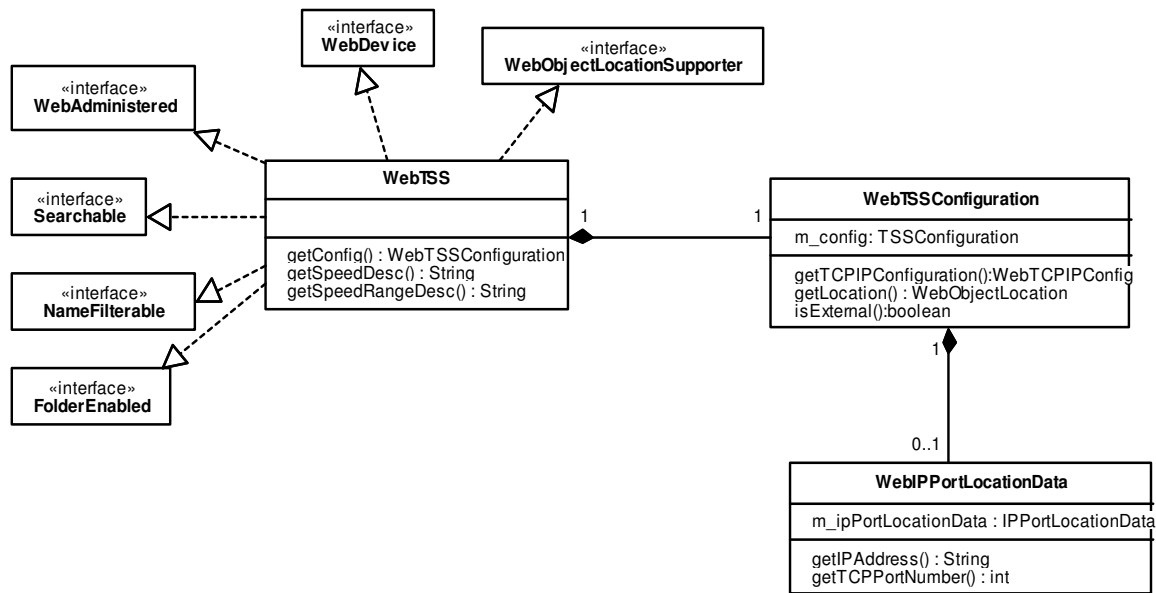


Figure 5-334. GUITSSDataClasses (Class Diagram)

5.40.1.1.1 FolderEnabled (Class)

This interface provides access to information about an object that can be stored in a folder.

5.40.1.1.2 NameFilterable (Class)

This java interface is implemented by classes which can be filter by name within the ObjectCache. A NameFilter object is passed into the ObjectCache to select NameFilterable objects in the cache.

5.40.1.1.3 Searchable (Class)

This interface allows objects to be searched for via a substring search.

5.40.1.1.4 WebAdministered (Class)

This interface allows the implementing class to be administered via the trader console pages.

5.40.1.1.5 WebDevice (Class)

This interface contains common functionality for CHART devices.

5.40.1.1.6 WebIPPortLocationData (Class)

This class wraps the IPPortLocationData IDL structure and provides accessor methods to get the data. This class has data for identifying a TCP/IP address and port.

5.40.1.1.7 WebObjectLocationSupporter (Class)

This interface allows common processing for objects supporting an ObjectLocation via the WebObjectLocation wrapper class..

5.40.1.1.8 WebTSS (Class)

This class wraps the TransportationSystemSensor CORBA interface, caches data, and provides access to the cached data.

5.40.1.1.9 WebTSSConfiguration (Class)

This class wraps the TSSConfiguration IDL structure and provides accessors for easy access to the data.

5.41 Chartlite.data.trafficevents-data

5.41.1 Classes

5.41.1.1 chartlite.data.trafficevents_classes (Class Diagram)

This diagram shows the main wrapper class used for storing traffic event related data in the cache.

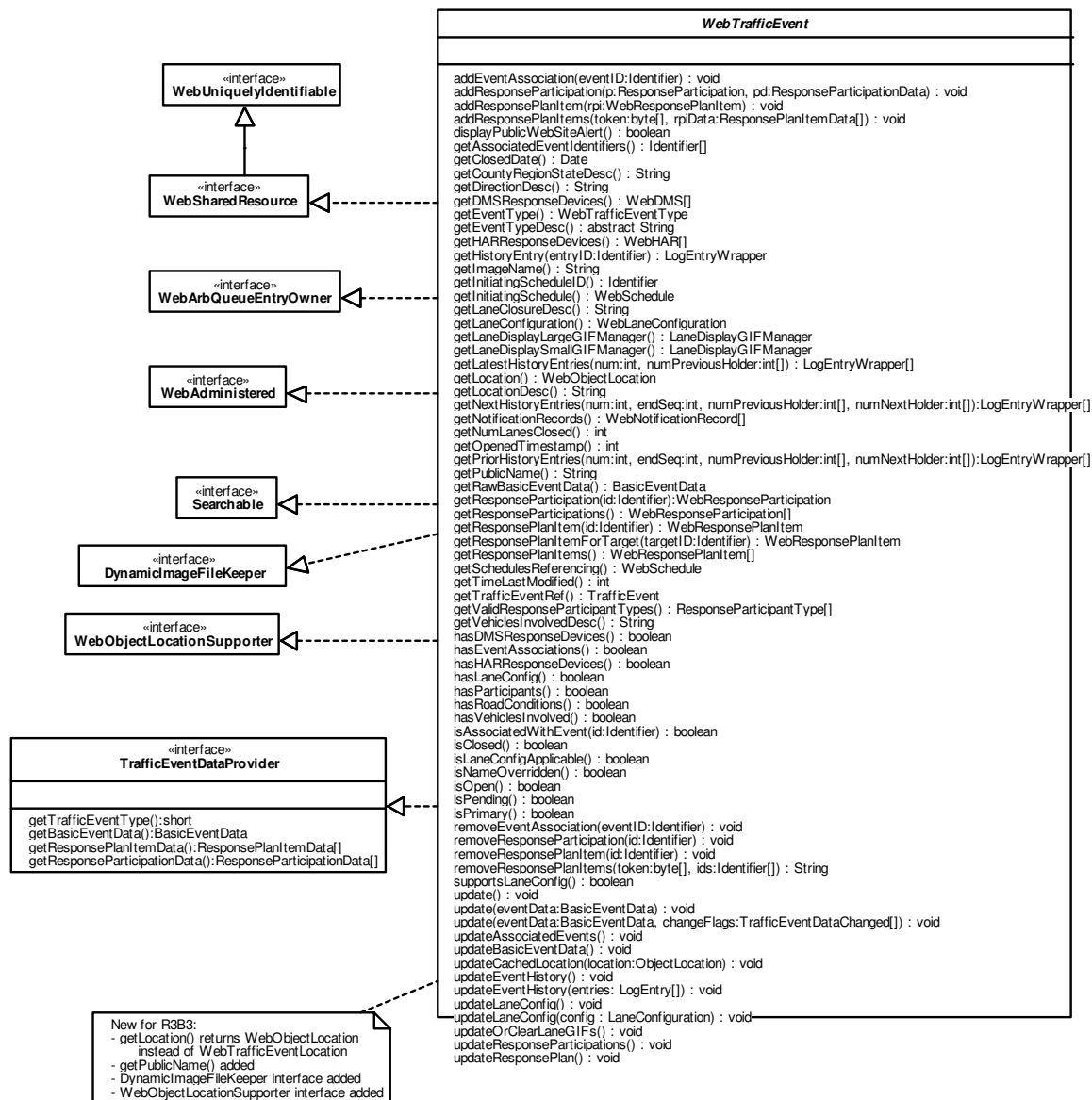


Figure 5-335. chartlite.data.trafficevents_classes (Class Diagram)

5.41.1.1.1 DynamicImageFileKeeper (Class)

This interface allows an object to keep dynamic image files from being deleted by the DynImageCleanupTask, which periodically deletes files that are no longer needed.

5.41.1.1.2 Searchable (Class)

This interface allows objects to be searched for via a substring search.

5.41.1.1.3 TrafficEventDataProvider (Class)

This interface is implemented by classes that can provide data from a traffic event. This interface exists because traffic event data may be accessible in different forms depending on where the ActionExecutionGroup (and related classes) are being used. For example, the data for a traffic event may be accessible via a cache of traffic event data or via a CORBA object reference.

5.41.1.1.4 WebAdministered (Class)

This interface allows the implementing class to be administered via the trader console pages.

5.41.1.1.5 WebArbQueueEntryOwner (Class)

This interface specifies methods to be implemented by all objects that may place entries on an arbitration queue.

5.41.1.1.6 WebObjectLocationSupporter (Class)

This interface allows common processing for objects supporting an ObjectLocation via the WebObjectLocation wrapper class..

5.41.1.1.7 WebSharedResource (Class)

This interface is implemented by any GUI-side wrapper objects representing CHART shared resources in the system, corresponding to the SharedResource IDL interface.

5.41.1.1.8 WebTrafficEvent (Class)

This class represents a TrafficEvent object in the system and caches its data for fast access. It provides accessor methods to get the cached data, in addition to auxiliary methods.

5.41.1.1.9 WebUniquelyIdentifiable (Class)

This interface provides functionality for GUI objects that represent UniquelyIdentifiable objects as defined in the IDL.

5.41.1.2 chartlite.data.trafficevents_event_type_classes (Class Diagram)

This diagram shows classes used to cache traffic event related data in the CHART GUI servlet.

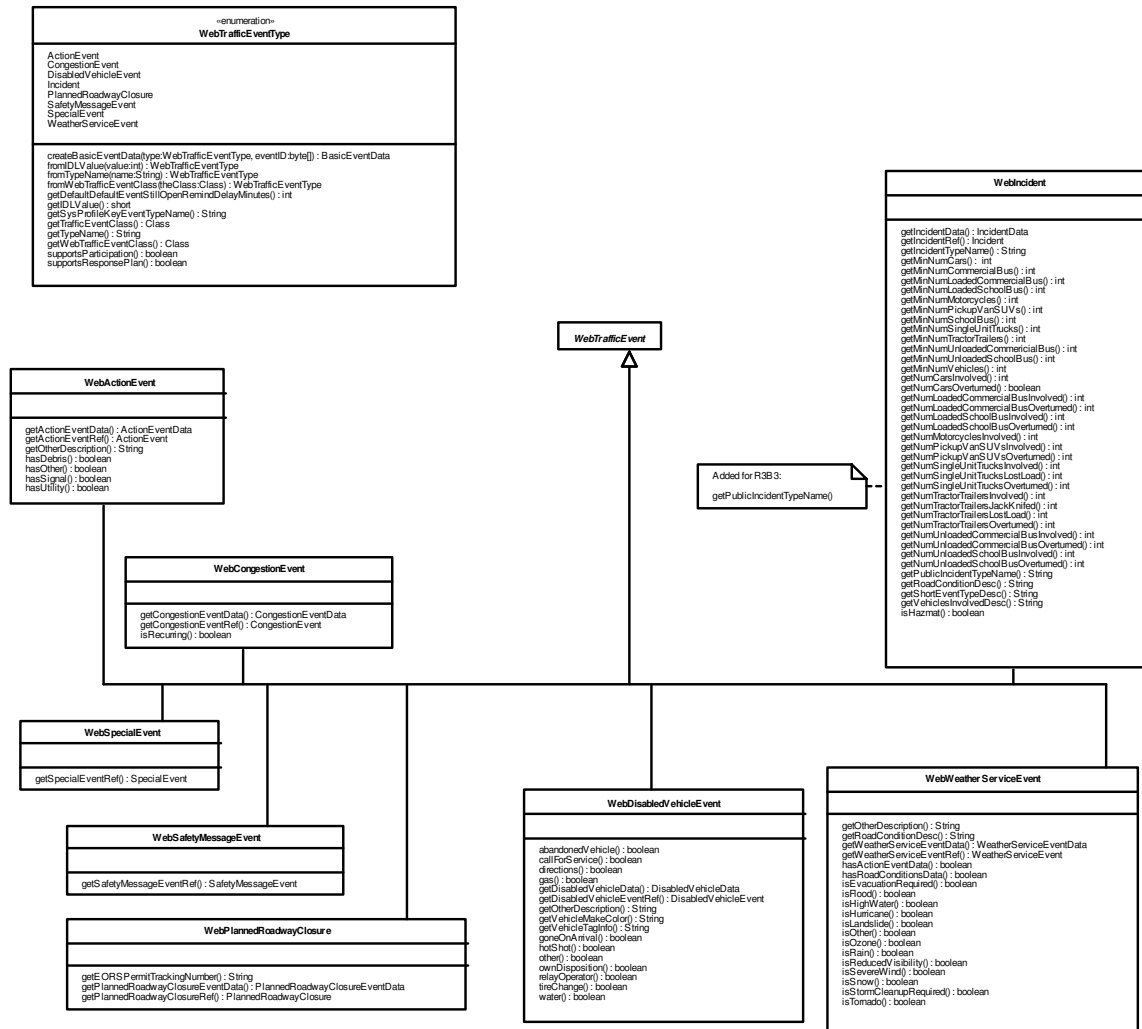


Figure 5-336. chartlite.data.trafficevents_event_type_classes (Class Diagram)

5.41.1.2.1 WebActionEvent (Class)

This class is a wrapper for a CORBA ActionEvent that allows it to be cached and to be accessed within Velocity templates.

5.41.1.2.2 WebCongestionEvent (Class)

This class is a wrapper for a CORBA CongestionEvent that allows it to be cached and to be accessed from within Velocity templates.

5.41.1.2.3 WebDisabledVehicleEvent (Class)

This class is a wrapper for a CORBA DisabledVehicleEvent that allows it to be cached and to be accessed from within a Velocity template.

5.41.1.2.4 WebIncident (Class)

This class is a wrapper for a CORBA Incident that allows it to be cached and to be accessed from within a Velocity template.

5.41.1.2.5 WebPlannedRoadwayClosure (Class)

This class is a wrapper for a CORBA PlannedRoadwayClosure that allows it to be cached and to be accessed from within a Velocity template.

5.41.1.2.6 WebSafetyMessageEvent (Class)

This class is a wrapper for a CORBA SafetyMessageEvent that allows it to be cached and to be accessed from within a Velocity template.

5.41.1.2.7 WebSpecialEvent (Class)

This class is a wrapper for a CORBA SpecialEvent that allows it to be cached and to be accessed from within a Velocity template.

5.41.1.2.8 WebTrafficEvent (Class)

This class represents a TrafficEvent object in the system and caches its data for fast access. It provides accessor methods to get the cached data, in addition to auxiliary methods.

5.41.1.2.9 WebTrafficEventType (Class)

This enumeration contains the traffic event types.

5.41.1.2.10 WebWeatherServiceEvent (Class)

This class is a wrapper for a CORBA WeatherServiceEvent that allows it to be cached and accessed from within a Velocity template.

5.41.1.3 chartlite.data.trafficevents_misc_classes (Class Diagram)

This diagram shows miscellaneous classes related to traffic events.

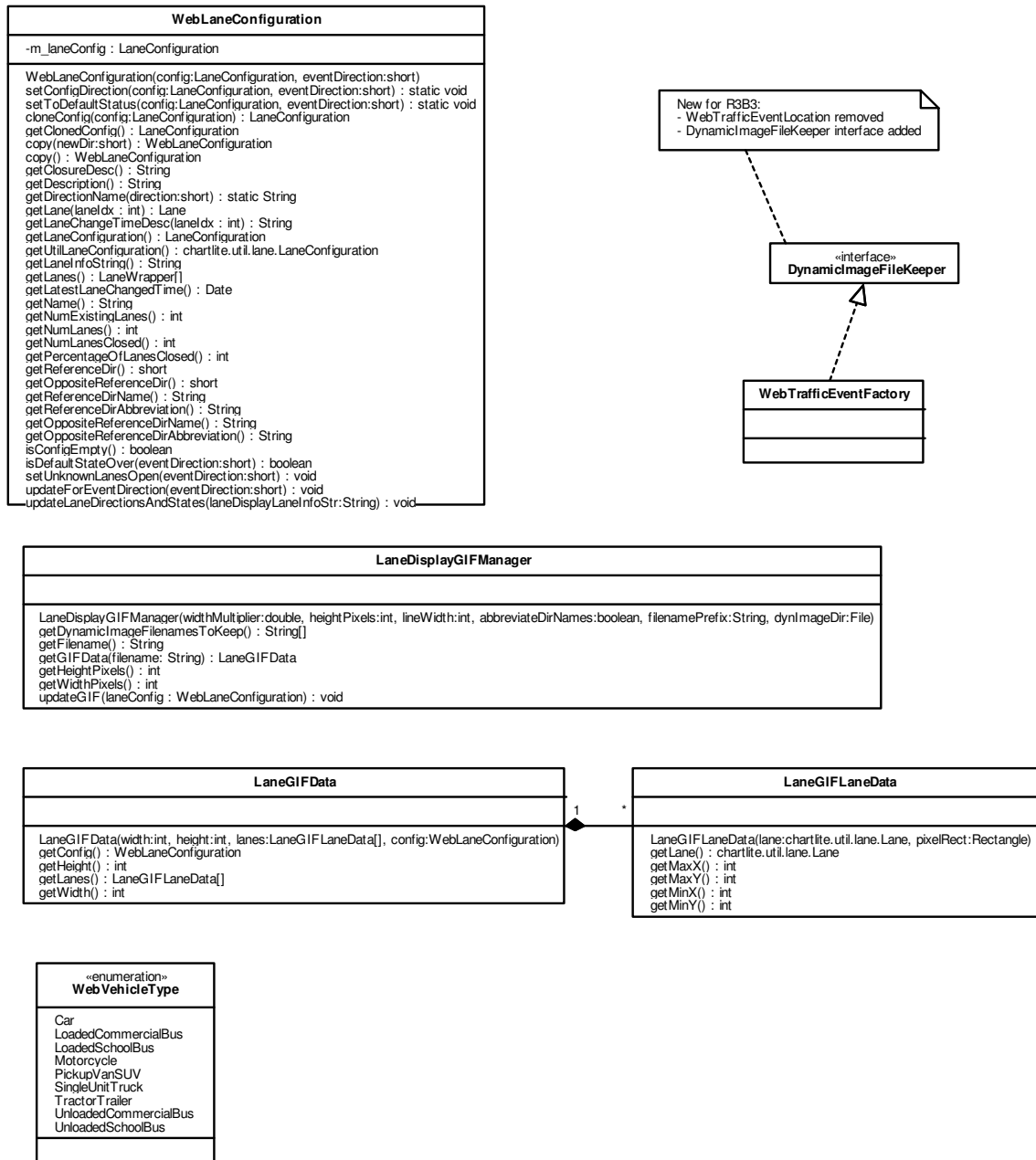


Figure 5-337. chartlite.data.trafficevents_misc_classes (Class Diagram)

5.41.1.3.1 DynamicImageFileKeeper (Class)

This interface allows an object to keep dynamic image files from being deleted by the DynImageCleanupTask, which periodically deletes files that are no longer needed.

5.41.1.3.2 LaneDisplayGIFManager (Class)

This class manages a GIF file representation of a lane configuration. The configuration may be updated, which would cause a new GIF file to be created.

5.41.1.3.3 LaneGIFData (Class)

This class contains metadata for a single instance of a GIF file, making it easy to create an image map for the file via Velocity.

5.41.1.3.4 LaneGIFLaneData (Class)

This class represents a single lane within a single instance of a GIF file. It is used when building an image map.

5.41.1.3.5 WebLaneConfiguration (Class)

This class wraps a LaneConfiguration structure and provides auxiliary methods for getting and manipulating the data.

5.41.1.3.6 WebTrafficEventFactory (Class)

This class represents the TrafficEventFactory object in the server and wraps it to provide faster access to cached data.

5.41.1.3.7 WebVehicleType (Class)

This enumeration lists the vehicle types that can be recorded in an Incident event

5.42 Chartlite.data.plans-data

5.42.1 Classes

5.42.1.1 plans_data_classes (Class Diagram)

This diagram shows classes related to filtering response plans/items.

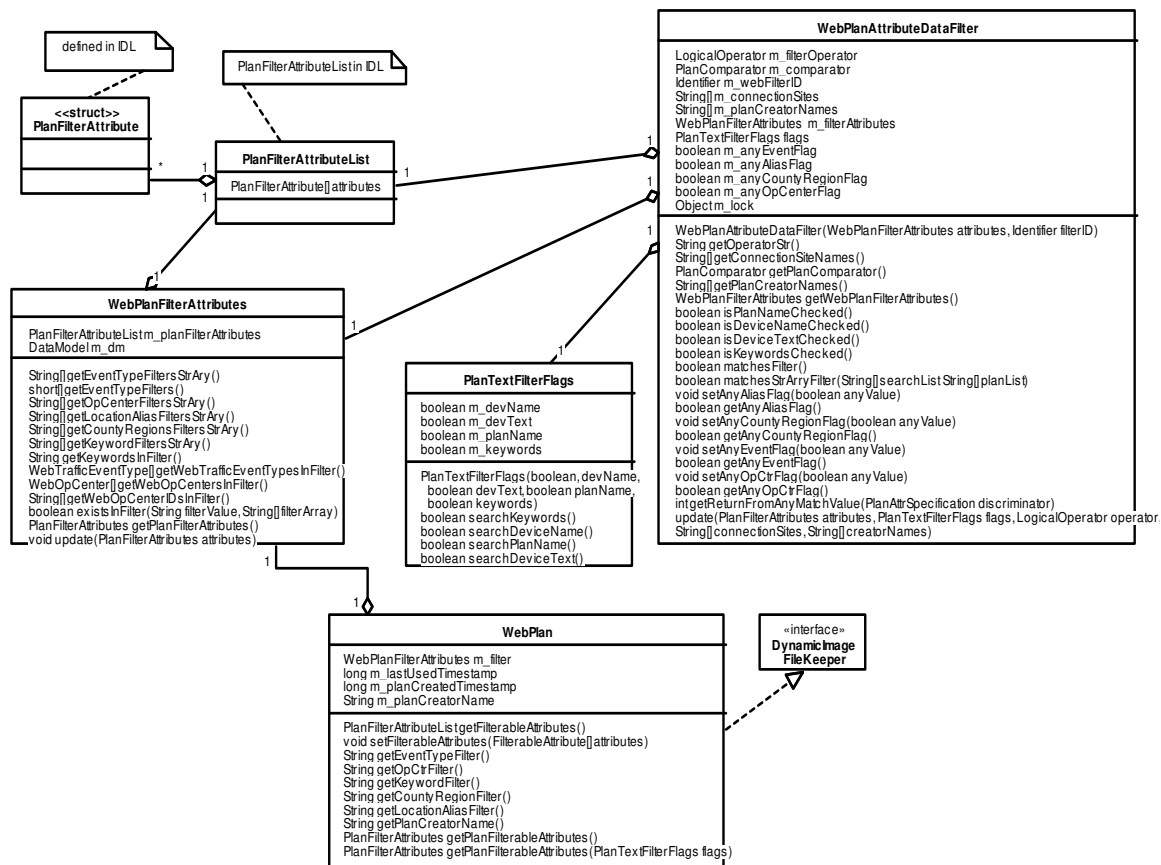


Figure 5-338. plans_data_classes (Class Diagram)

5.42.1.1.1 DynamicImage FileKeeper (Class)

This interface allows an object to keep dynamic image files from being deleted by the **DynImageCleanupTask**, which periodically deletes files that are no longer needed.

5.42.1.1.2 PlanFilterAttributeList (Class)

5.42.1.1.3 PlanTextFilterFlags (Class)

5.42.1.1.4 WebPlan (Class)

5.42.1.1.5 WebPlanAttributeDataFilter (Class)

5.42.1.1.6 WebPlanFilterAttributes (Class)

5.42.1.2 plans_data_classes (Class Diagram)

This diagram shows classes related to filtering response plans/items.

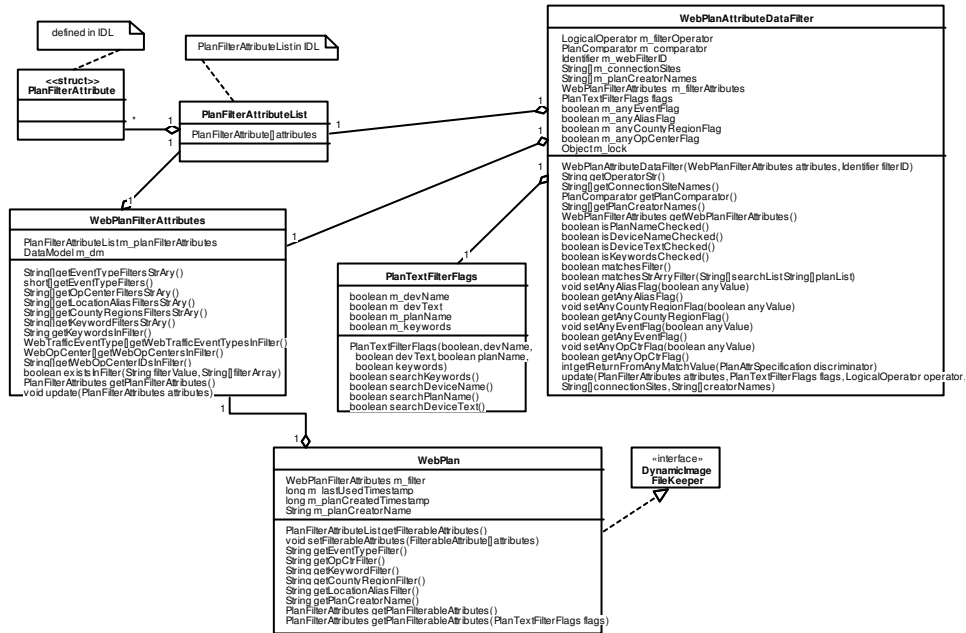


Figure 5-339. plans_data_classes (Class Diagram)

5.42.1.2.1 <<struct>> PlanFilterAttribute (Class)

5.42.1.2.2 DynamicImage FileKeeper (Class)

This interface allows an object to keep dynamic image files from being deleted by the DynImageCleanupTask, which periodically deletes files that are no longer needed.

5.42.1.2.3 PlanFilterAttributeList (Class)

5.42.1.2.4 PlanTextFilterFlags (Class)

5.42.1.2.5 WebPlan (Class)

5.42.1.2.6 WebPlanAttributeDataFilter (Class)

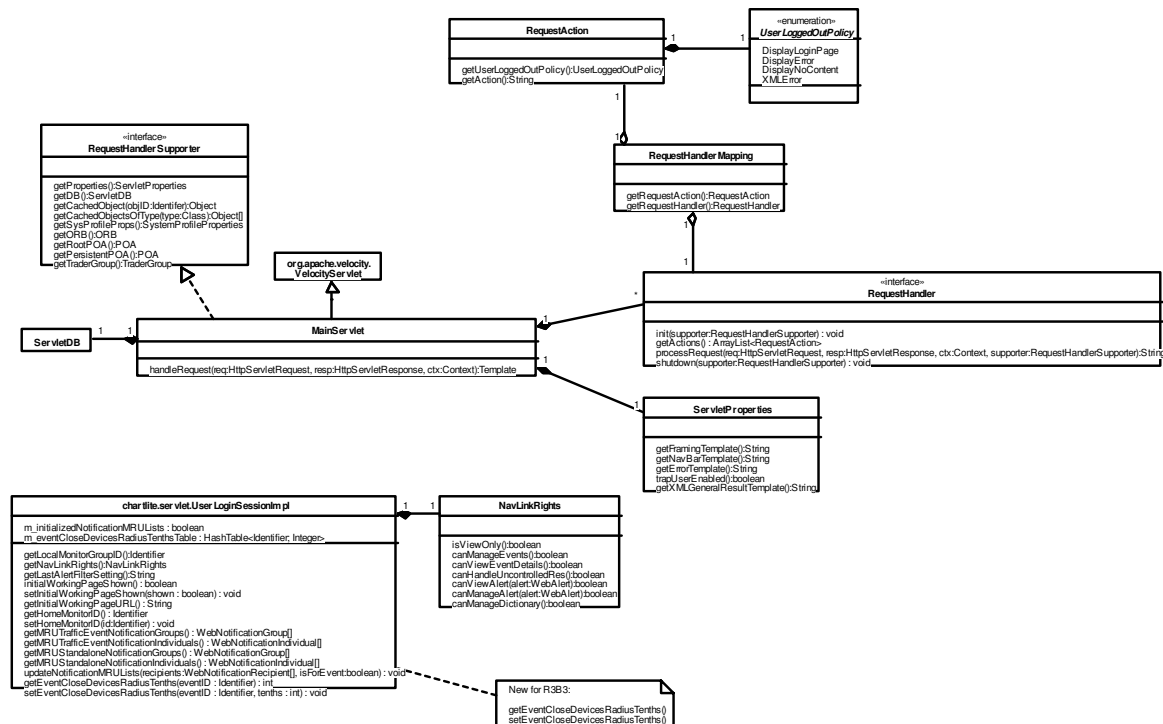
5.42.1.2.7 WebPlanFilterAttributes (Class)

5.43 Chartlite.servlet

5.43.1 Classes

5.43.1.1 ServletBaseClasses (Class Diagram)

This diagram shows classes related to the base CHART GUI servlet.



5-340 ServletBaseClasses (Class Diagram)

5.43.1.1.1 chartlite.servlet.UserLoginSessionImpl (Class)

This class is used to store information about the logged in user. It is also the implementation of the UserLoginSession CORBA interface that can be called from the server to ensure the user is still logged in, send them an instant message, or force the user to become logged out.

5.43.1.1.2 MainServlet (Class)

This class is the main class of the servlet. It handles all requests and dispatches them to the appropriate request handler. It also acts as a RequestHandlerSupporter, which is passed to each request handler to help them process requests.

5.43.1.1.3 NavLinkRights (Class)

This class provides user rights checking for the servlet. It contains a user's token and provides easy to use methods that can check the presence of functional rights, combinations of rights, or even rights that are specific to the object the user wishes to use.

5.43.1.1.4 org.apache.velocity. VelocityServlet (Class)

The base class for the Velocity template engine. This template engine is used to provide dynamic content from the CHART GUI Servlet. The web pages are code in templates using velocity specific macros. The code in the servlet loads data that will be shown on the page into a velocity Context, and this VelocityServlet class is used to merge the content with the template to create HTML for the browser to display.

5.43.1.1.5 RequestAction (Class)

This class contains information about an action that can be invoked via a request handler. The action parameter is specified in the URL as the "action" parameter, or as the last part of the servlet path. The user logged out policy specifies what the servlet should do if this action is requested when the user is logged out.

5.43.1.1.6 RequestHandler (Class)

This interface specifies methods that are to be implemented by classes that are used to process requests.

5.43.1.1.7 RequestHandlerMapping (Class)

This class provides a mapping between an action and the request handler used to process a request for that action.

5.43.1.1.8 RequestHandlerSupporter (Class)

This interface is implemented by any class that can provide access to objects or methods that are helpful to request handlers.

5.43.1.1.9 ServletDB (Class)

This class is used by the CHART GUI servlet to access CHART GUI specific data that is stored in the database.

5.43.1.1.10ServletProperties (Class)

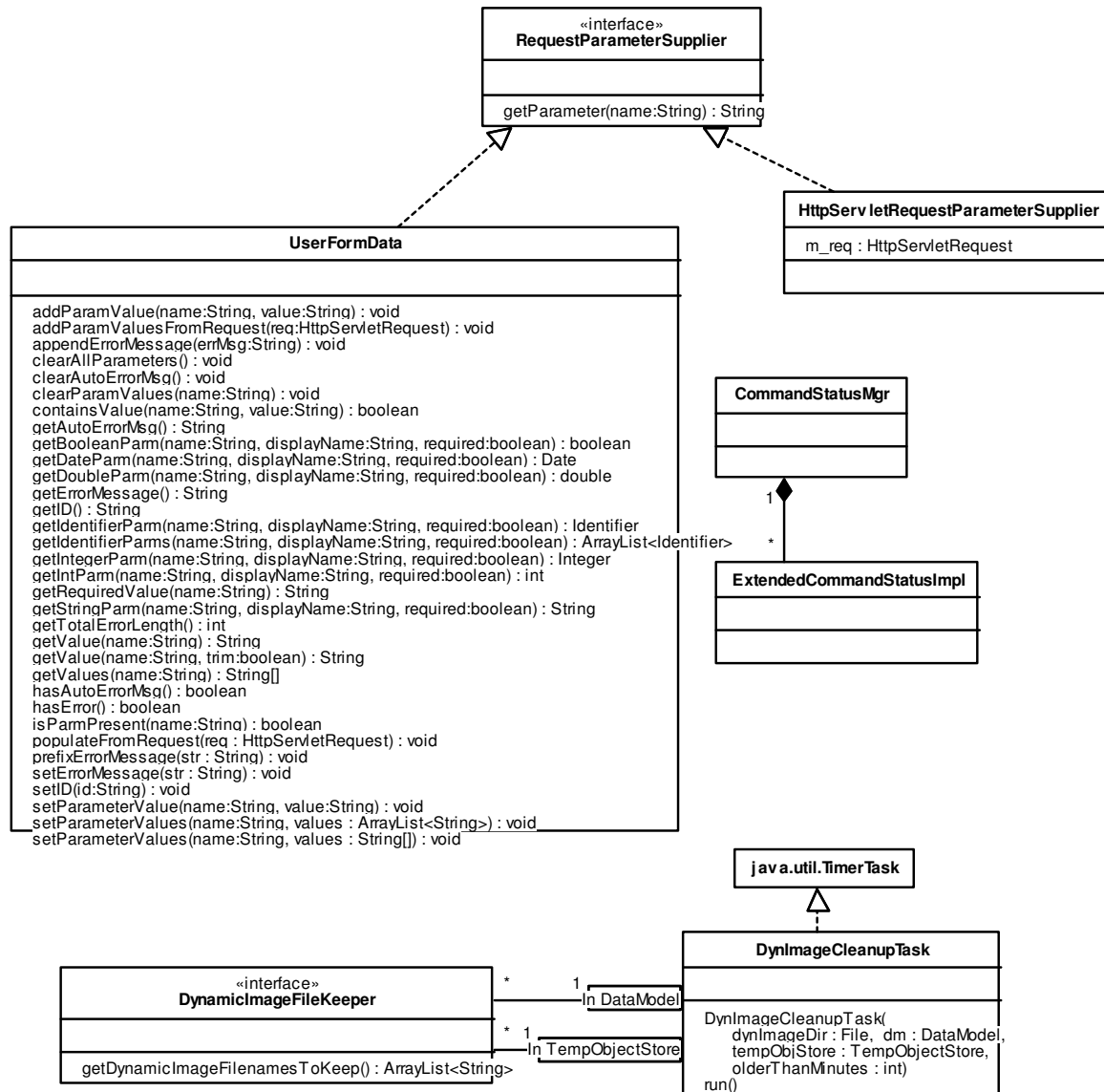
This class provides access to properties defined in the chart gui's properties file.

5.43.1.1.11UserLoggedOutPolicy (Class)

This enumeration specifies the types of actions that may be specified for responding to a request that is received when the user is logged out.

5.43.1.2 ServletMiscClasses (Class Diagram)

This diagram shows miscellaneous classes used within the servlet.



5-341. ServletMiscClasses (Class Diagram)

5.43.1.2.1 CommandStatusMgr (Class)

This class manages command status objects served by the GUI.

5.43.1.2.2 DynamicImageFileKeeper (Class)

This interface allows an object to keep dynamic image files from being deleted by the `DynImageCleanupTask`, which periodically deletes files that are no longer needed.

5.43.1.2.3 DynImageCleanupTask (Class)

This class periodically cleans up dynamic image files in the dynamic images directory that are no longer needed. The files to keep are maintained by objects in the DataModel and TempObjectStore that implement the DynamicImageFileKeeper interface.

5.43.1.2.4 ExtendedCommandStatusImpl (Class)

This is an abstract class that is extended by classes that implement the ExtendedCommandStatus CORBA interface. It handles the basic implementation required of a command status, and leaves the implementation of updateAny() and completedAny() to the derived classes.

5.43.1.2.5 HttpServletRequestParameterSupplier (Class)

This class implements the RequestParameterSupplier interface to provide parameters from the HttpServletRequest.

5.43.1.2.6 java.util.TimerTask (Class)

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

5.43.1.2.7 RequestParameterSupplier (Class)

This interface allows parameter values to be queried. It is used to provide a common interface for getting parameters from the HttpServletRequest or from the UserFormData.

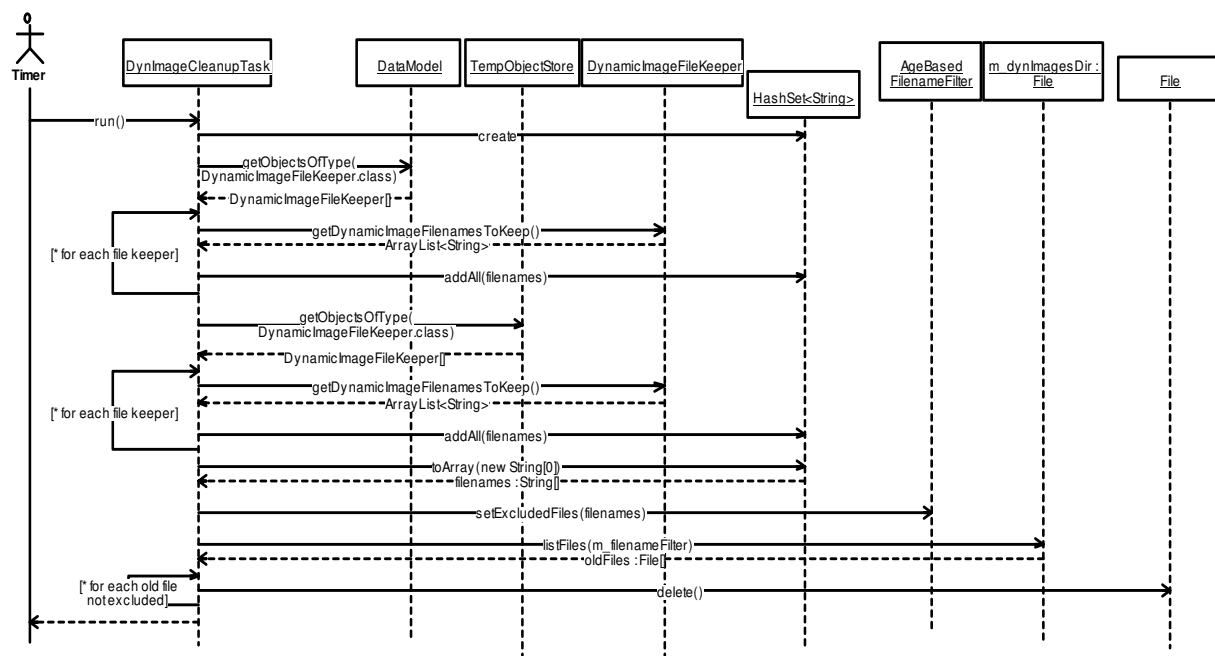
5.43.1.2.8 UserFormData (Class)

This class is used to store form data between requests while a user is editing a complex form, and provides convenience methods for parsing the values from the request.

5.43.2 Sequence Diagrams

5.43.2.1 DynImageCleanupTask:run (Sequence Diagram)

This diagram shows how old dynamic image files are cleaned up. The DynImageCleanupTask is called periodically to run, and it finds all objects implementing the DynamicImageFileKeeper interface in the DataModel and the TempObjectStore. It asks them for the files they want to keep, and sets those files as excluded files in the AgeBasedFilenameFilter. The filter is then called to get the files to delete, and the files are deleted.



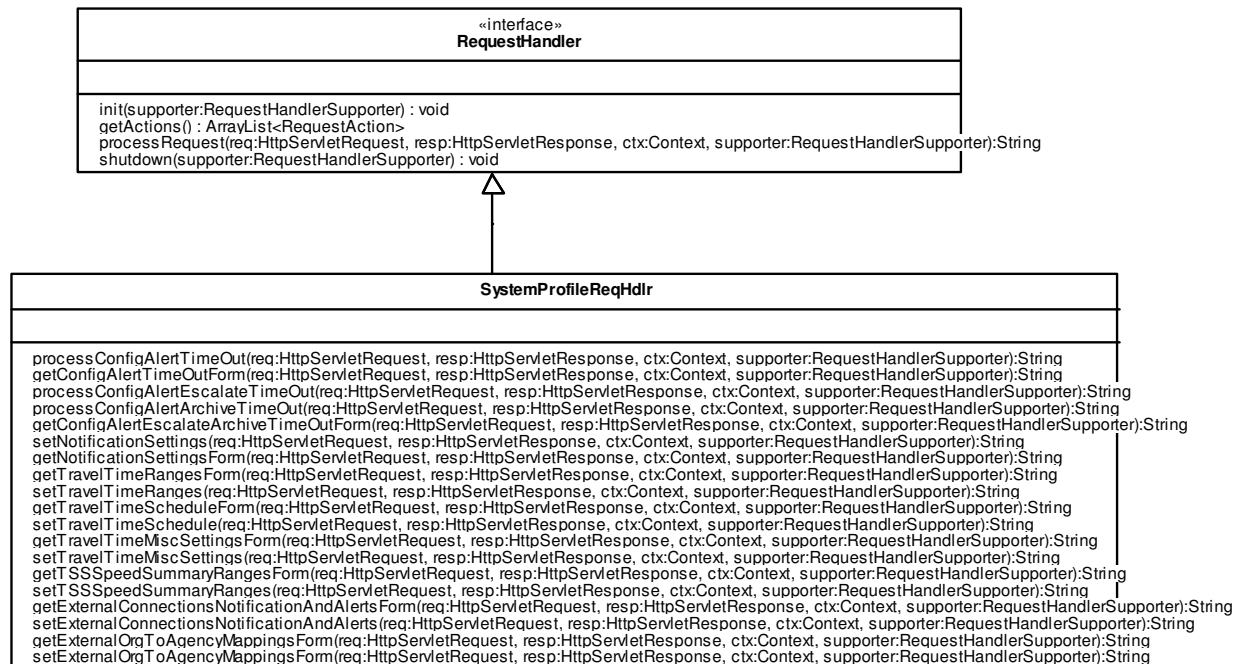
5-342. DynImageCleanupTask:run (Sequence Diagram)

5.44 Chartlite.servlet.usermgmt

5.44.1 Class Diagrams

5.44.1.1 chartlite.servlet.usermgmt.systemProfile_classes (Class Diagram)

This diagram shows CHART GUI servlet classes related to the system profile.



5-343. chartlite.servlet.usermgmt.systemProfile_classes (Class Diagram)

5.44.1.1.1 RequestHandler (Class)

This interface specifies methods that are to be implemented by classes that are used to process requests.

5.44.1.1.2 SystemProfileReqHdlr (Class)

This class is a request handler that processes requests related to the system profile.

5.44.2 Sequence diagrams

5.44.2.1 SystemProfileReqHdlr:addDMSMsgComboProps (Sequence Diagram)

This diagram shows the processing that is performed in the addDMSMsgComboProps method of the SystemProfileReqHdlr class. This method is used to read the parameters passed from the DMS Message combination Rules form and store them into a Properties object that can be passed to SystemProfileProperties object to set the properties. This method exists prior to R3B3 and is upgraded in R3B3 to handle message combinations that result from 2 new message queue buckets (priorities), Toll Rate and Travel Time. The general pattern used by this method is to construct a property name using a prefix and IDL values for the two types of priorities being combined. The parameter associated with the check box that sets whether or not messages of those priorities can be combined is retrieved from the HttpServletRequest. If the parameter is present, that indicates the box checked, and otherwise the box is unchecked. The appropriate value (true or false) is stored in the Properties object that was passed to the method using the appropriate property name. In R3B3, the following parameters are added to the form and to this method:

canCombineUrgentWithTollRate canCombineUrgentWithTravelTime
canCombineIncidentWithTollRate canCombineIncidentWithTravelTime
canCombinePlannedClosureWithTollRate canCombinePlannedClosureWithTravelTime
canCombineTollRateWithTollRate canCombineTollRateWithTravelTime
canCombineTollRateWithCongestion canCombineTollRateWithSHAZAM
canCombineTravelTimeWithTravelTime canCombineTravelTimeWithCongestion
canCombineTravelTimeWithSHAZAM

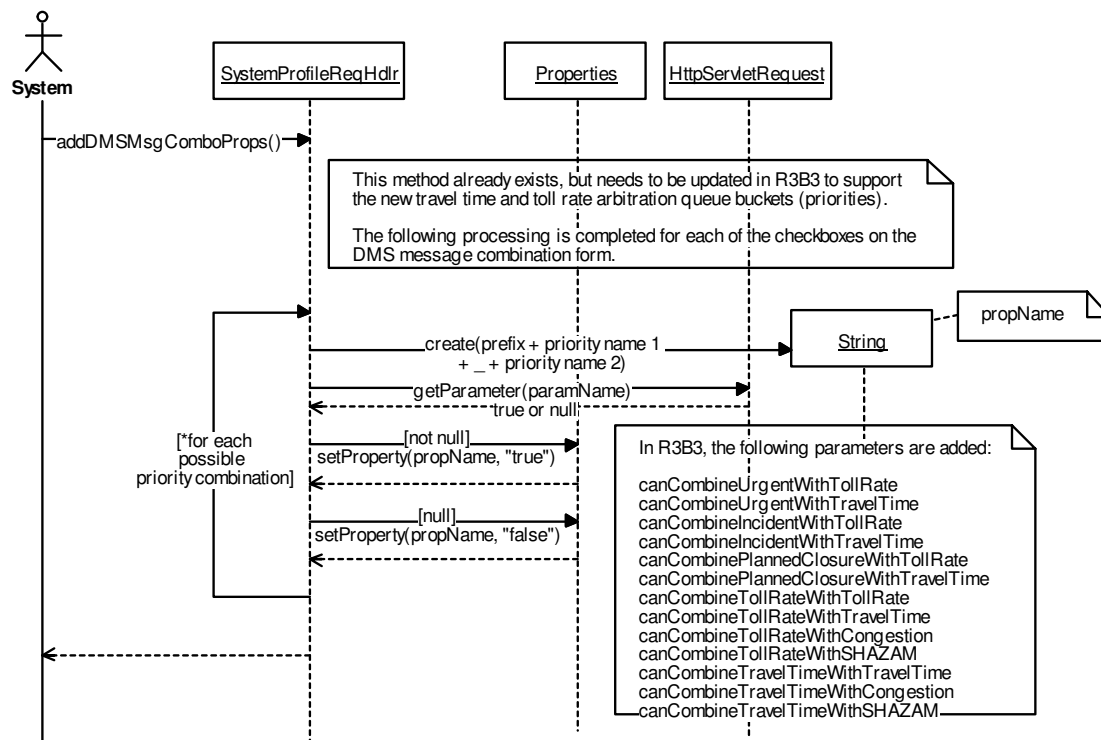


Figure 5-344. SystemProfileReqHdr:addDMSMsgComboProps (Sequence Diagram)

5.44.2.2 SystemProfileReqHdr:getExternalConnectionAlertAndNotificationSettingsForm (Sequence Diagram)

This diagram shows the processing that occurs when an administrator chooses to manage the External Connection Alert and Notification settings. The settings for each connection are retrieved from the system profile in JSON string format. Each JSON string is parsed to build a JSON object and placed in a JSON array which is ultimately added to the context.

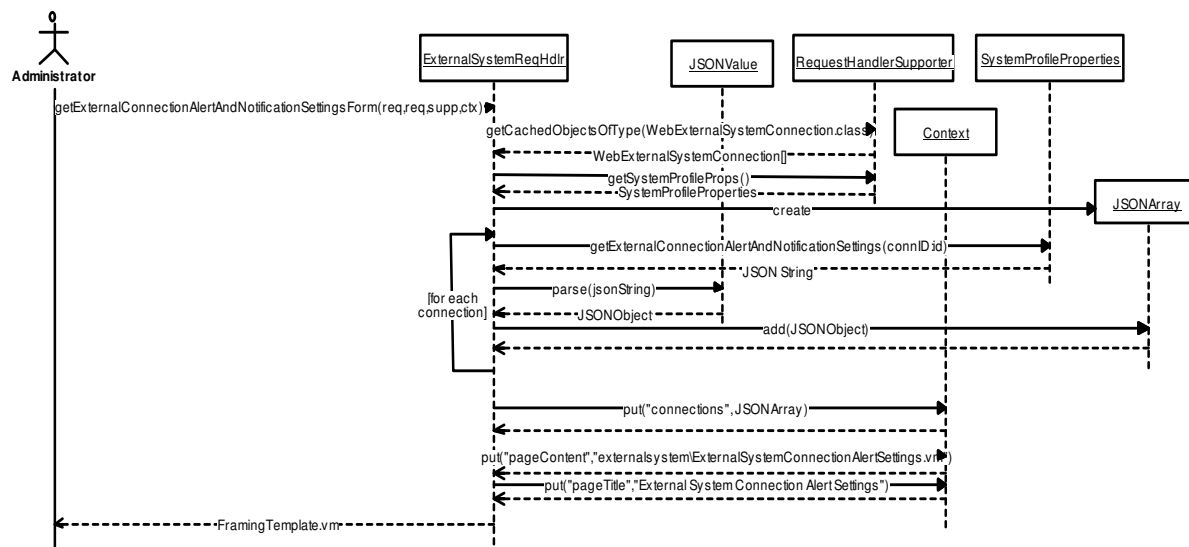


Figure 5-345.
SystemProfileReqHdr:getExternalConnectionAlertAndNotificationSettingsForm
(Sequence Diagram)

5.44.2.3 SystemProfileReqHdlr:getExternalOrgToAgencyMappingsForm (Sequence Diagram)

This diagram show the processing that occurs when an administrator displays the external agencies to oranzations mapping form.

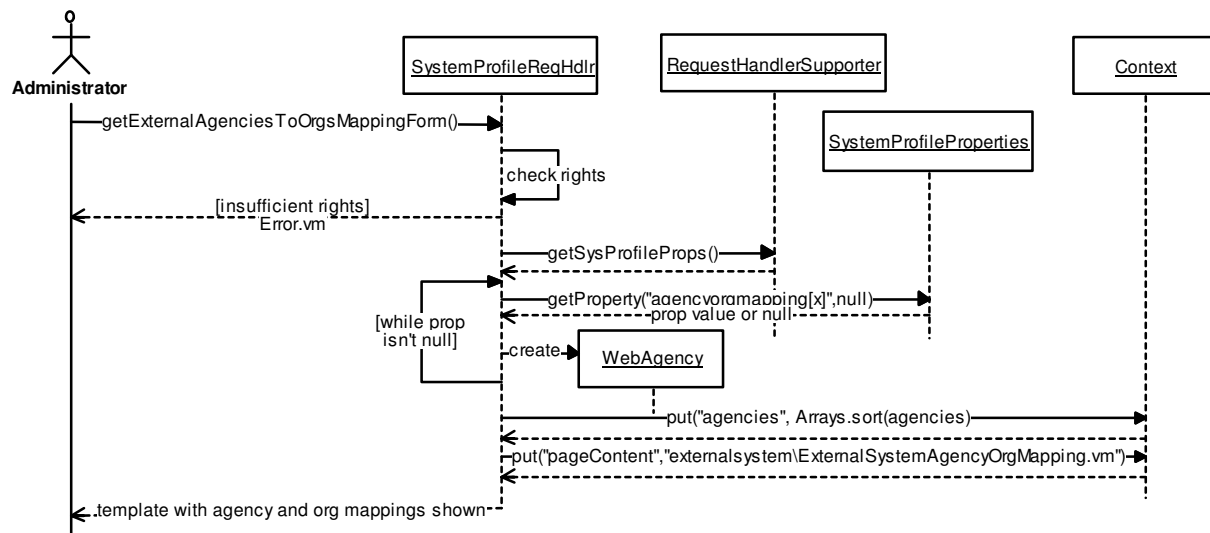


Figure 5-346. SystemProfileReqHdlr:getExternalOrgToAgencyMappingsForm (Sequence Diagram)

5.44.2.4 SystemProfileReqHdlr:getTSSSpeedSummaryRangesForm (Sequence Diagram)

This diagrams show the processing that occurs when an administrator has chosen to display the TSS Speed Summary Ranges Form.

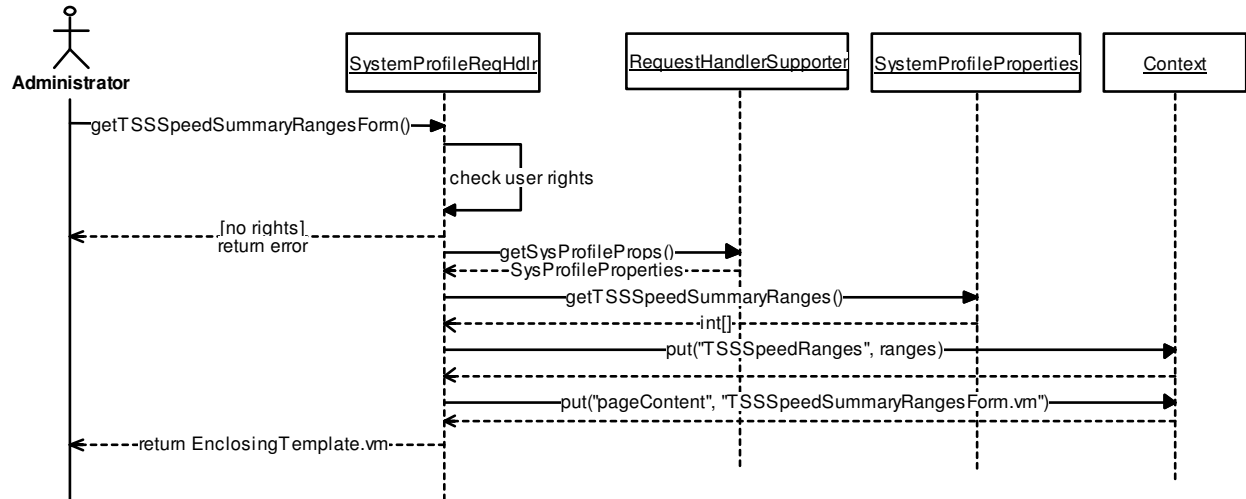


Figure 5-347. SystemProfileReqHdlr:getTSSSpeedSummaryRangesForm (Sequence Diagram)

5.44.2.5 SystemProfileReqHdr:getTravelTimeMiscSettingsForm (Sequence Diagram)

This diagram shows the processing that is performed when the administrator chooses to display the form used to change miscellaneous travel time settings. If they have not been granted the right to change the system profile, an error message is shown. Otherwise, the settings are retrieved from the system profile and placed in a UserFormData object, which is placed into the context so that the values can be used to pre-populate the form. The form is then displayed to the user.

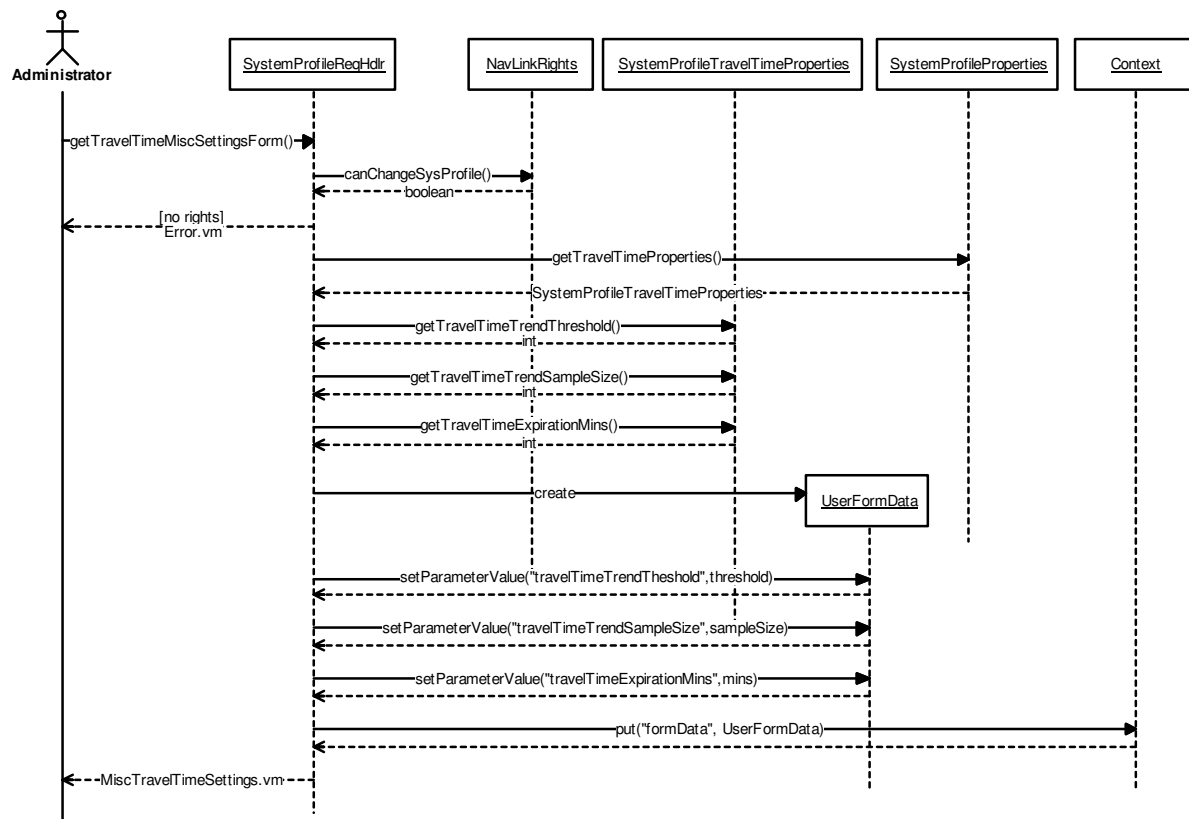


Figure 5-348. SystemProfileReqHdr:getTravelTimeMiscSettingsForm (Sequence Diagram)

5.44.2.6 SystemProfileReqHdlr:getTravelTimeRangesForm (Sequence Diagram)

This diagram shows the processing that is performed when the administrator chooses to display the form used to set the travel time range definitions. If they have not been granted the right to change the system profile, an error message is shown. Otherwise, the system profile properties object is called to obtain a SystemProfileTravelTimeProperties object, which is a wrapper that provides convenience methods for travel time related properties. This object is called to retrieve the current travel time range definitions from the system profile. The ranges are stored in the system profile as a string that uses JSON notation; this string is passed to the constructor of TravelTimeRange which parses the string and stores each range as a TravelTimeRangeDef object. This list of TravelTimeRangeDef objects is obtained from the TravelTimeRange object and traversed. The values from each TravelTimeRangeDef object are placed into a UserFormData object, using indexed parameter names, with the index changing for each object processed. The number of ranges is also placed in the form data. This form data object will be used by the form to populate its form fields. This technique is used to allow the form to be easily re-populated if an error is detected when the form is submitted. This form data object is put into the velocity context and the TravelTimeRangeSettings form is displayed to the administrator.

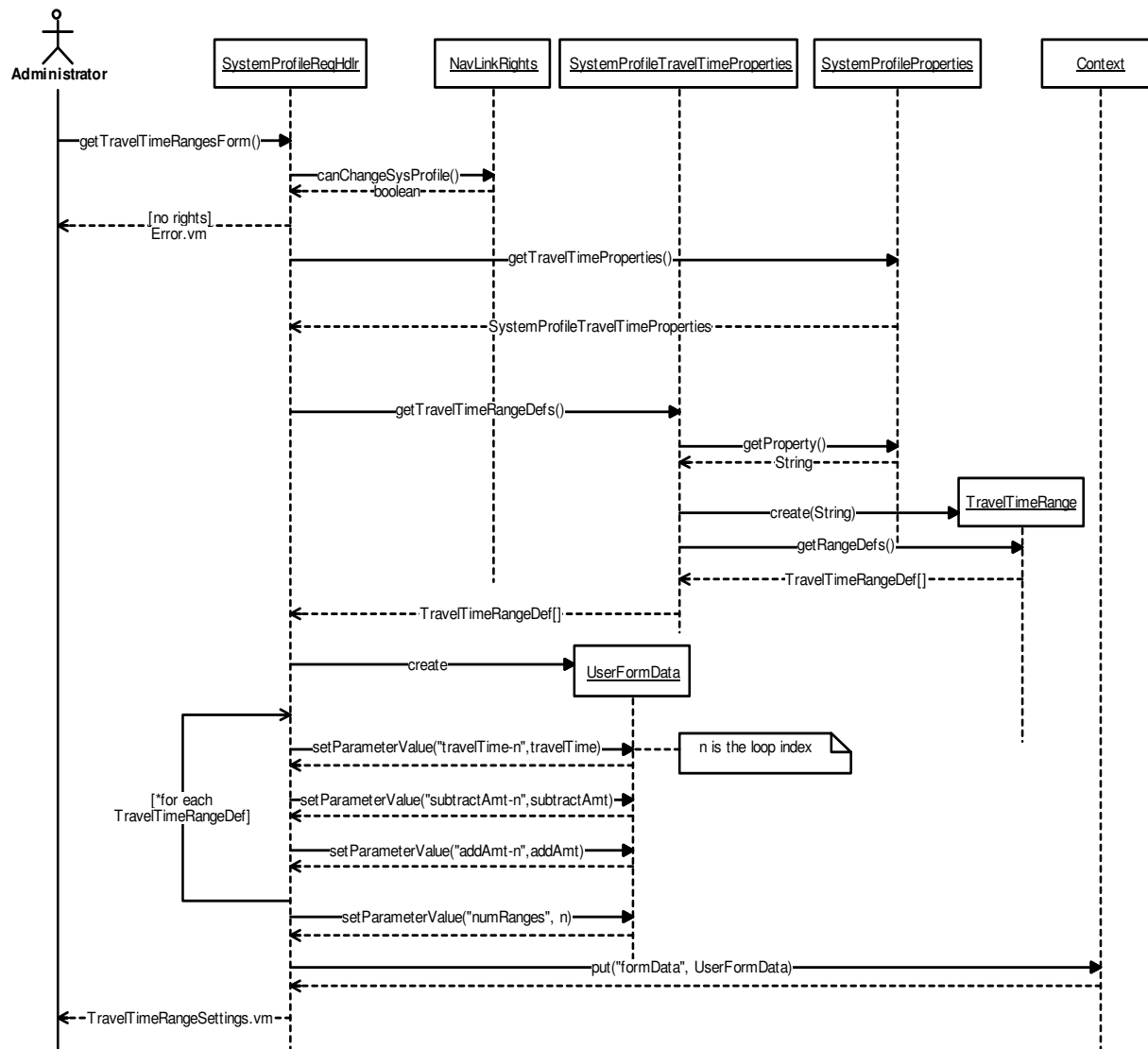


Figure 5-349. SystemProfileReqHdlr:getTravelTimeRangesForm (Sequence Diagram)

5.44.2.7 SystemProfileReqHdlr:getTravelTimeScheduleForm (Sequence Diagram)

This diagram shows the processing that is performed when the administrator chooses to display the form used to set the default (system-wide) travel time message display schedule. If they have not been granted the right to change the system profile, an error message is shown. Otherwise, the setting that specifies if specific time ranges are to be used (the alternative is display travel time messages 24x7) is retrieved from the system profile properties, as is the current list of specific time ranges. The time ranges are stored in the system profile as a JSON encoded string and a utility method is used to convert the JSON string into an array of HHMMRange objects. A UserFormData object is constructed and the settings retrieved from the system profile properties are stored so that they may be used to pre-populate the form. For the time ranges, the values are stored using an index appended to the parameter name since there can be multiple time ranges. The form data is placed in the context and the travel time message schedule form is displayed to the user.

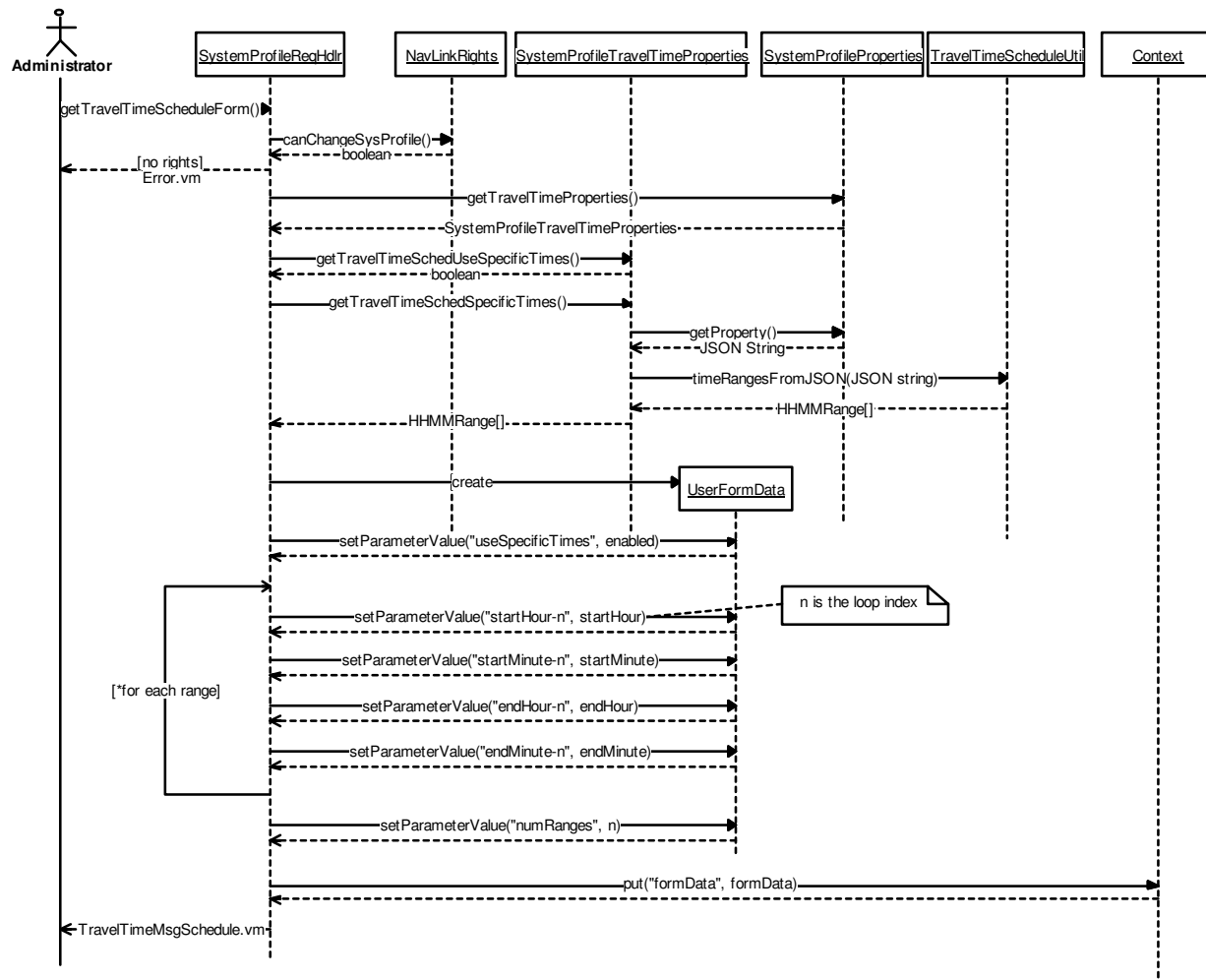


Figure 5-350. SystemProfileReqHdlr:getTravelTimeScheduleForm (Sequence Diagram)

5.44.2.8 SystemProfileReqHdlr:setExternalAgencyToOrgMappings (Sequence Diagram)

This diagram shows the processing that occurs when an adminsitator sets the external to agency to CHART organization mappings

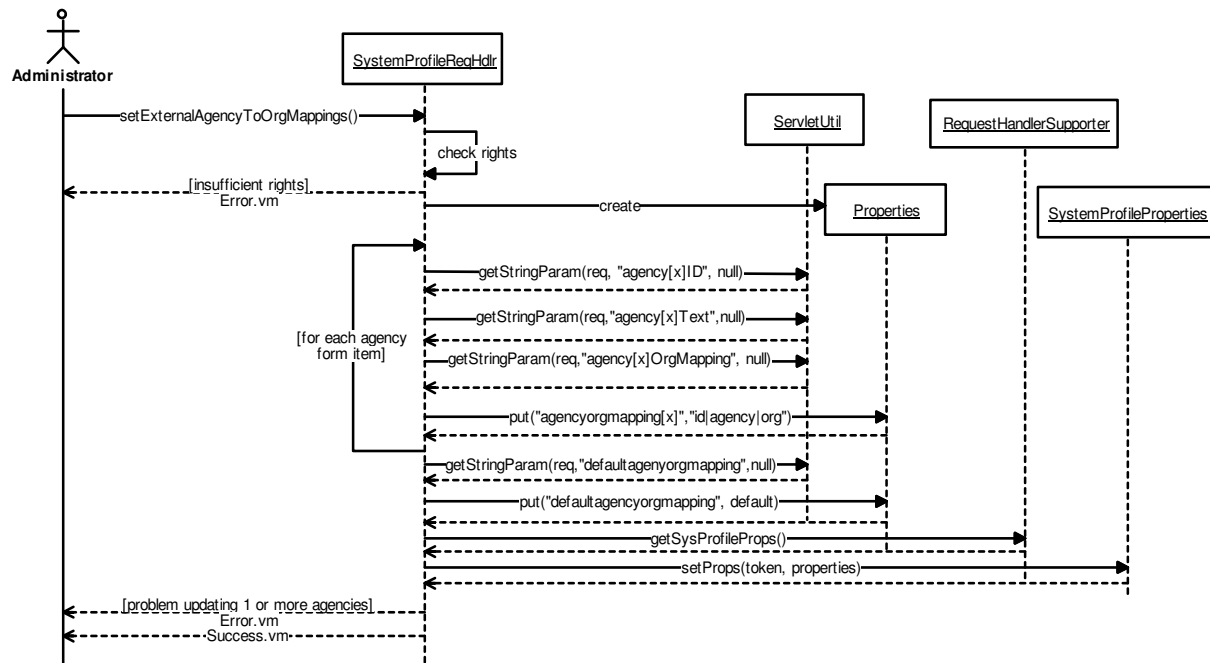


Figure 5-351. SystemProfileReqHdlr:setExternalAgencyToOrgMappings (Sequence Diagram)

5.44.2.9 SystemProfileReqHdlr:setExternalConnectionAlertAndNotificationSettingsForm (Sequence Diagram)

This diagram shows the processing that occurs when the administrator submits the alert and notifications settings for external connections.

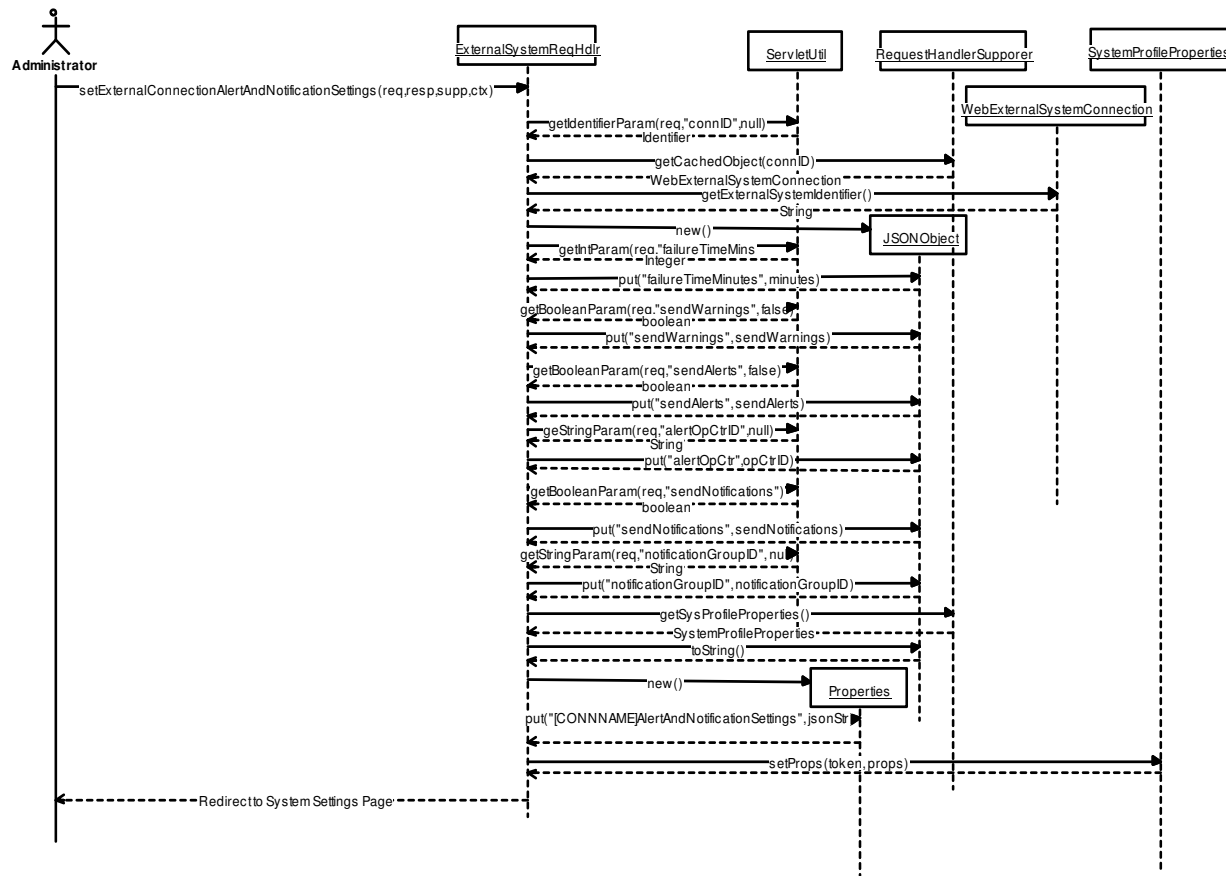


Figure 5-352.
SystemProfileReqHdlr:setExternalConnectionAlertAndNotificationSettingsForm
(Sequence Diagram)

5.44.2.10 SystemProfileReqHdlr:setTSSSpeedSummaryRanges (Sequence Diagram)

This diagrams shows the processing that occurs when an administrator submits the form for specifying TSS speed summary ranges.

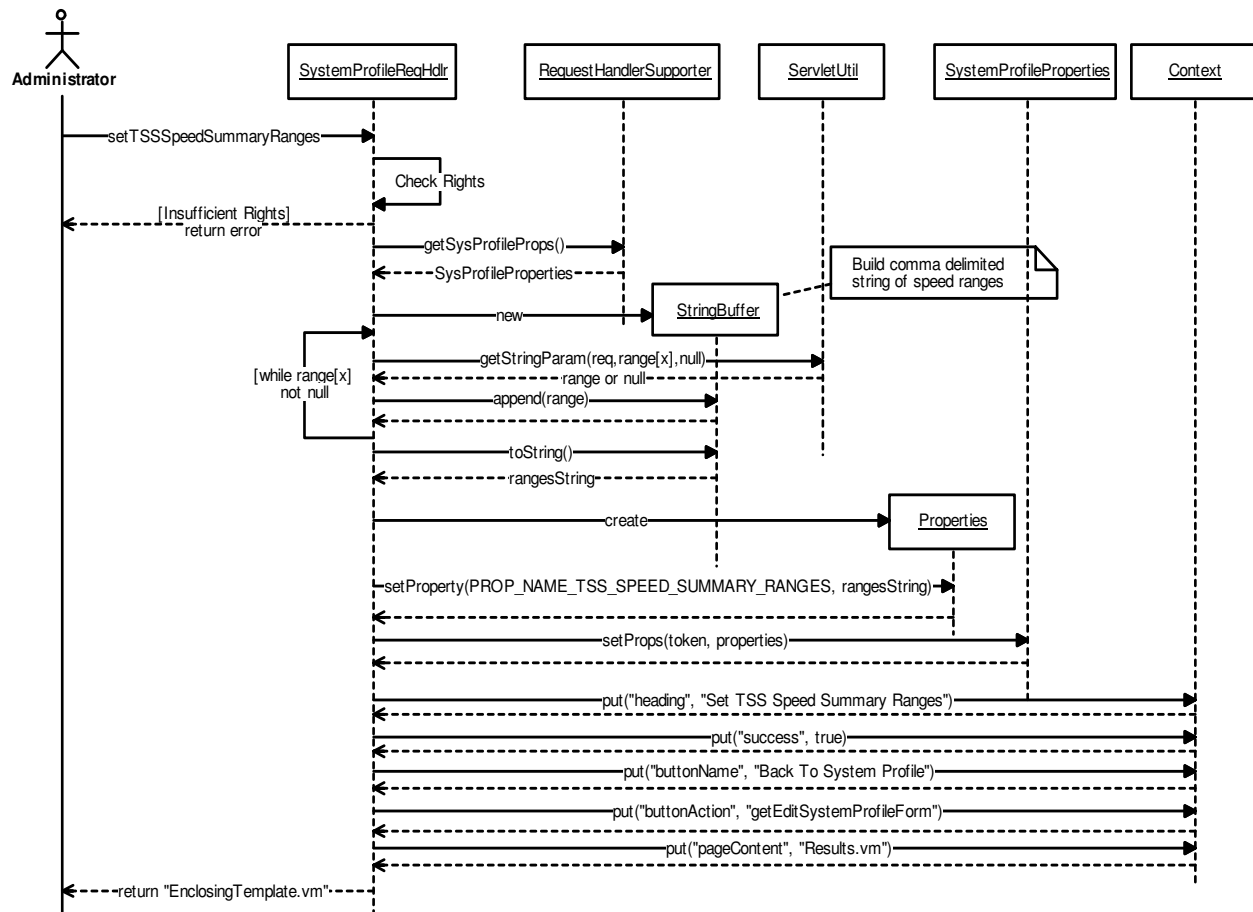


Figure 5-353. SystemProfileReqHdlr:setTSSSpeedSummaryRanges (Sequence Diagram)

5.44.2.11 SystemProfileReqHdlr:setTravelTimeMiscSettings (Sequence Diagram)

This diagram shows the processing that is performed when the administrator chooses to submit the form used to set the miscellaneous travel time settings. If they have not been granted the right to change the system profile, an error message is shown. Otherwise, a UserFormData object is constructed and populated with the request parameters. Each expected request parameter is retrieved from the UserFormData and validation is performed. Any detected errors result in the form being redisplayed with the user's entries and an error message at the top. If there are no errors a Properties object is constructed and populated with the data from the form. The Properties object is passed to the SystemProfileProperties.setProps() method to store the settings in the server, and a confirmation page is shown to the user.

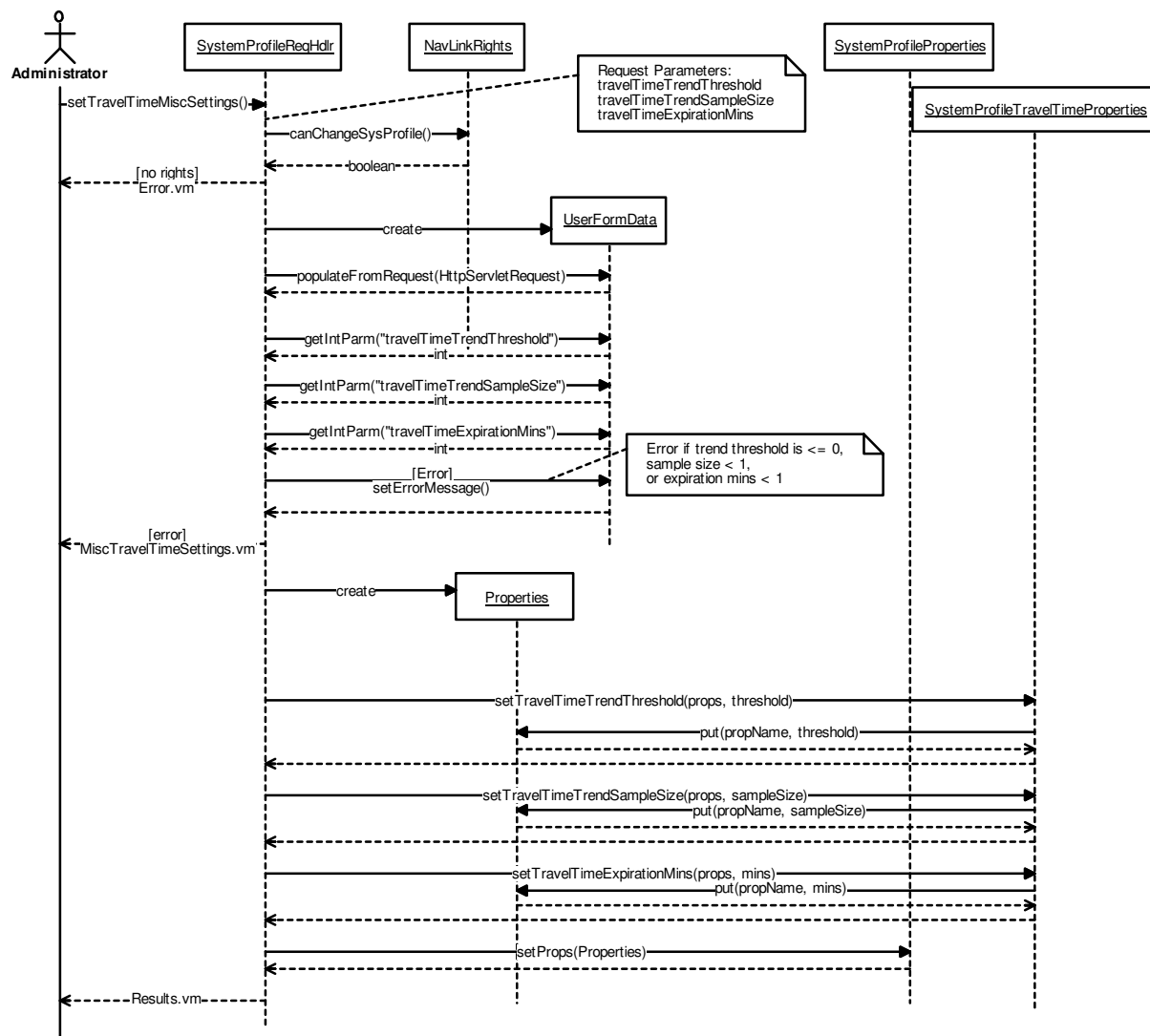


Figure 5-354. SystemProfileReqHdlr:setTravelTimeMiscSettings (Sequence Diagram)

5.44.2.12 SystemProfileReqHdlr:setTravelTimeRanges (Sequence Diagram)

This diagram shows the processing that is performed when the administrator chooses to submit the form used to set the travel time range definitions. If they have not been granted the right to change the system profile, an error message is shown. Otherwise, a UserFormData object is constructed and populated with the request parameters. A loop is executed using an increasing index to retrieve the indexed request parameters for each defined travel time range. Each set of parameters is used to construct a TravelTimeRangeDef object, and each of these objects is stored in an ArrayList. The loop ends when a parameter with the current index is not found. A Properties object is constructed and populated using the static setTravelTimeRangeDets method of the SystemProfileTravelTimeProperties class. That class uses a TravelTimeRange object to create the JSON string that is stored in the system profile (all JSON code related to travel time ranges is located in the TravelTimeRange class). The Properties object is passed to the SystemProfileProperties.setProps() method to store the travel time range settings in the server, and a confirmation page is shown to the user.

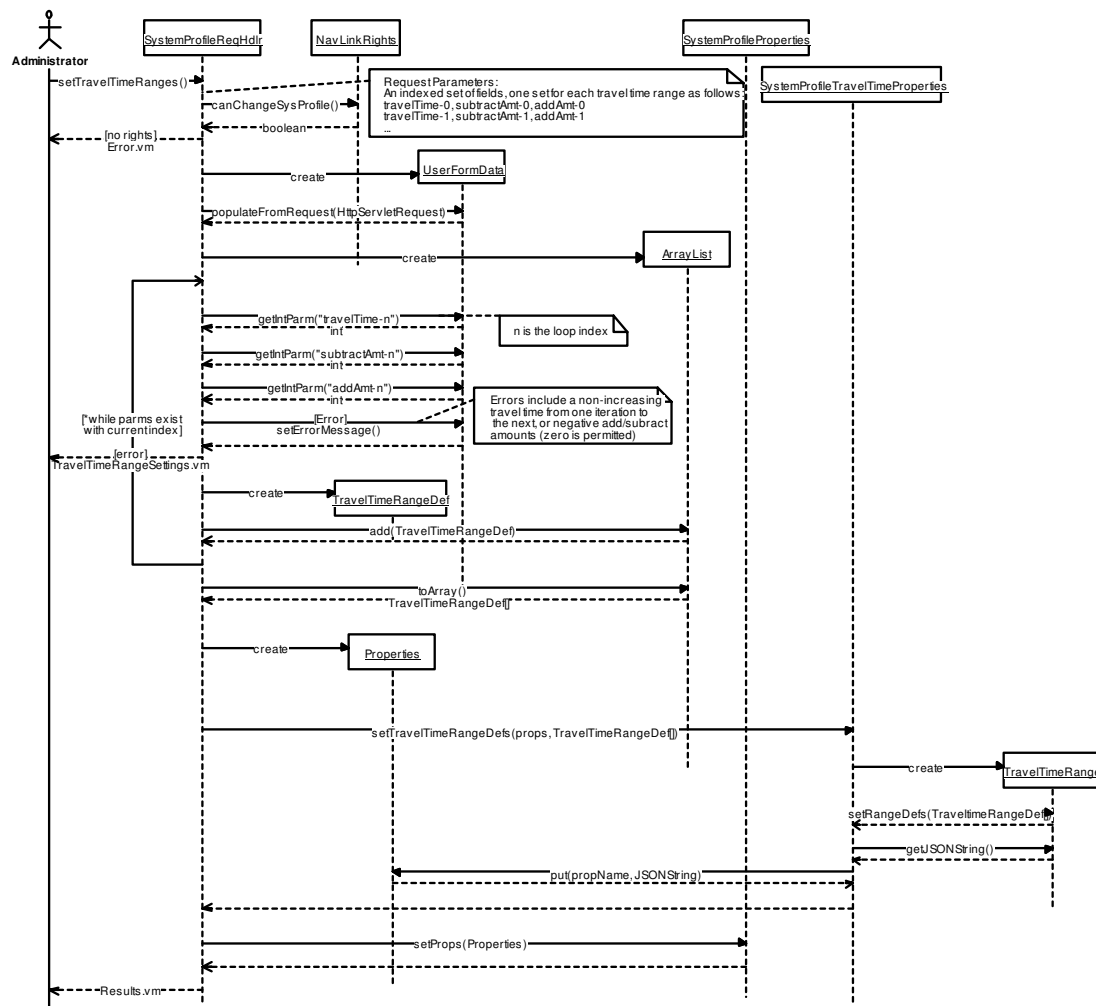


Figure 5-355. SystemProfileReqHdlr:setTravelTimeRanges (Sequence Diagram)

5.44.2.13 **SystemProfileReqHdlr:setTravelTimeSchedule (Sequence Diagram)**

This diagram shows the processing that is performed when the administrator chooses to submit the form used to set the system-wide travel time message display schedule. If they have not been granted the right to change the system profile, an error message is shown. Otherwise, a `UserFormData` object is constructed and populated with the request parameters. If the user has chosen to enter specific times when travel time messages may be displayed (rather than indicating they can be displayed 24x7), a loop is executed using an increasing index to retrieve the indexed request parameters for each defined time range. Each set of parameters is used to construct an `HHMMRange`, and each of these objects is stored in an `ArrayList`. The loop ends when a parameter with the current index is not found. A `Properties` object is constructed and populated with the selection of whether or not to use specific time ranges, and if specific time ranges are used the time ranges are added to the properties object after converting the list to a JSON array. Note that when the user chooses 24x7 (no specific time ranges), we leave the previous time ranges in-tact, allowing the user to easily enable them in the future if desired without having to re-enter all of the ranges. The `Properties` object is passed to the `SystemProfileProperties.setProps()` method to store the settings in the server, and a confirmation page is shown to the user.

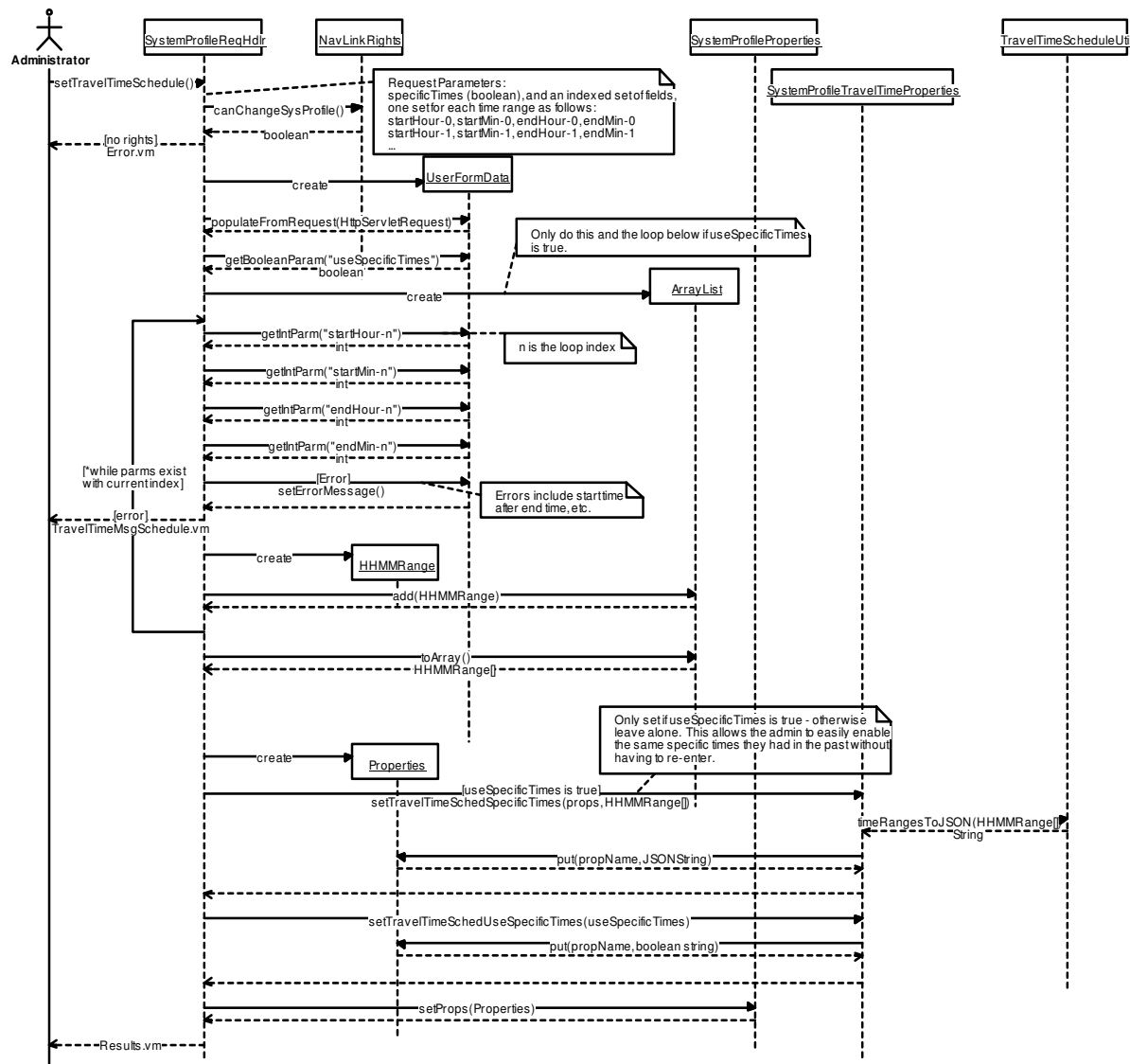


Figure 5-356. SystemProfileReqHdlr:setTravelTimeSchedule (Sequence Diagram)

5.45 Chartlite.servlet.tss

5.45.1 Classes

5.45.1.1 chartlite.servlet.tss_classes (Class Diagram)

This diagram shows CHART GUI servlet classes related to traffic sensor signs.

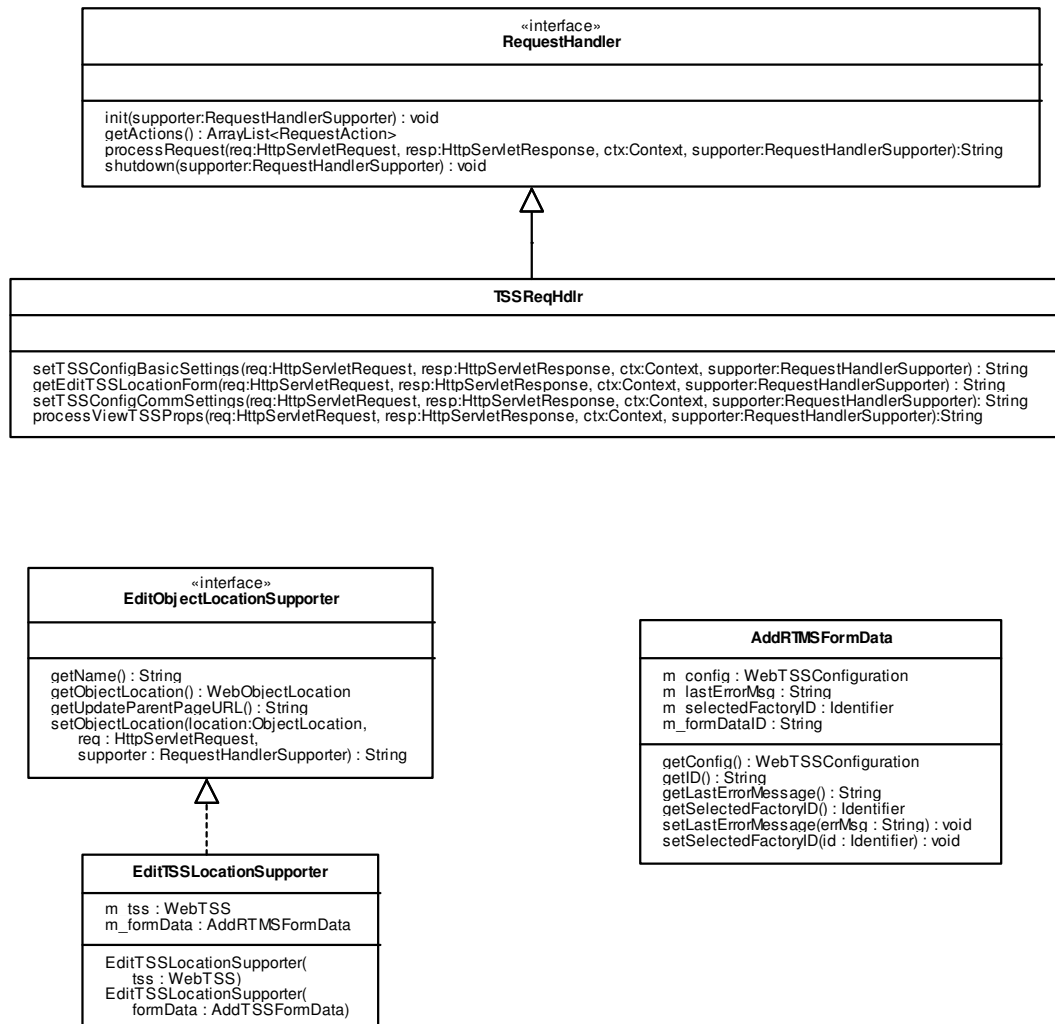


Figure 5-357. chartlite.servlet.tss_classes (Class Diagram)

5.45.1.1.1 AddRTMSFormData (Class)

This class represents the data in the Add RTMS form.

5.45.1.1.2 EditObjectLocationSupporter (Class)

This interface provides functionality allowing the location data to be edited. (For example, the target of the edited location may be an existing object, or it may be a form data object for creating a new object).

5.45.1.1.3 EditTSSLocationSupporter (Class)

This class is used to support editing the location of an existing or new TSS.

5.45.1.1.4 RequestHandler (Class)

This interface specifies methods that are to be implemented by classes that are used to process requests.

5.45.1.1.5 TSSReqHdlr (Class)

This class handles requests related to traffic sensor systems such as RTMS.

5.45.1.2 chartlite.servlet.tss_dynlist_classes (Class Diagram)

This diagram shows classes related to using TSS devices in dynamic lists.

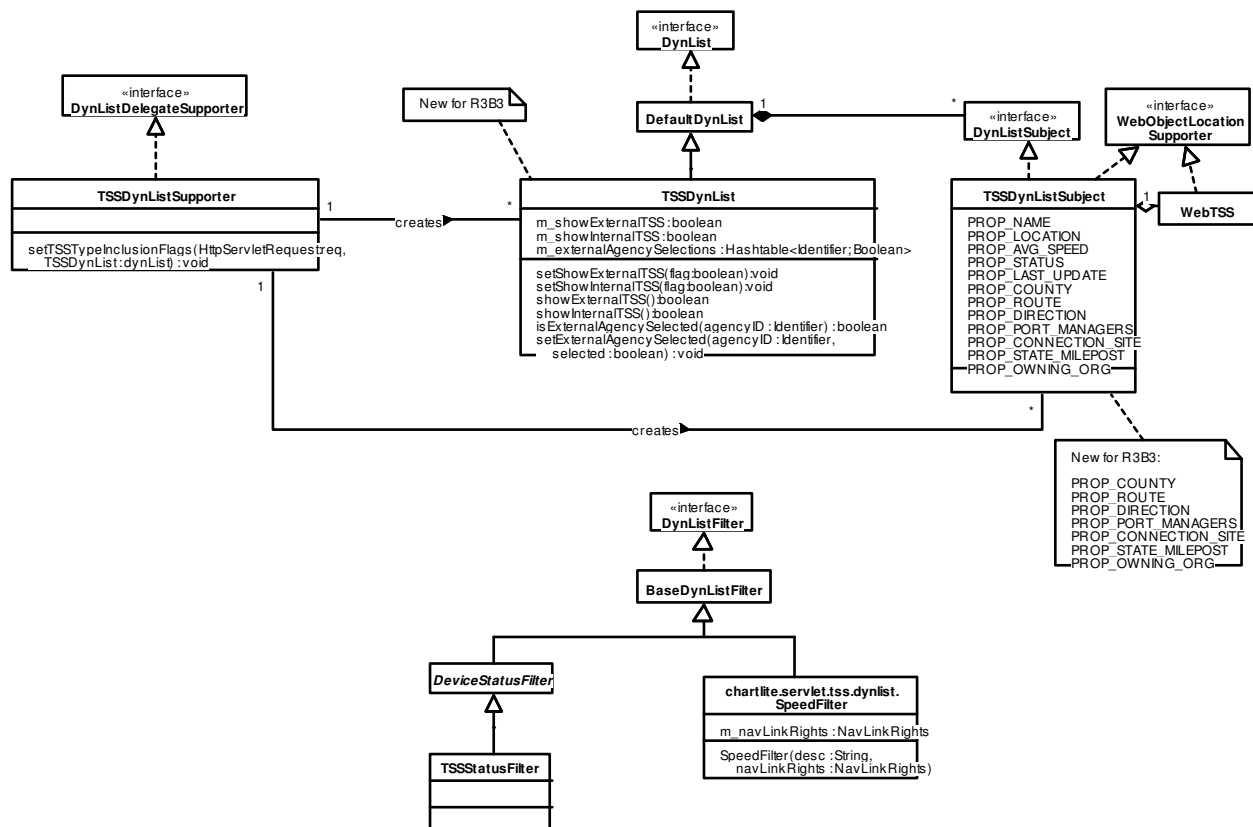


Figure 5-358. chartlite.servlet.tss_dynlist_classes (Class Diagram)

5.45.1.2.1 BaseDynListFilter (Class)

This abstract class provides a base implementation of the DynListFilter interface.

5.45.1.2.2 chartlite.servlet.tss.dynlist. SpeedFilter (Class)

This class is a filter for the TSS dynamic list that filters on the current speed value. For R3B3, it will examine the user's rights for each TSS to determine whether the user is allowed to view the average speed or speed ranges for a TSS, and will apply the filter based on the viewable data.

5.45.1.2.3 DefaultDynList (Class)

This class provides a default implementation of the DynList interface. It supports a collection of columns, a collection of global filters, and a collection of subjects. Filters in this list are treated additively - that is, a subject must pass all filters to be displayed.

5.45.1.2.4 DeviceStatusFilter (Class)

This class is a base filter for filtering on the device status. It must be extended to get the WebDevice from the DynListSubject object.

5.45.1.2.5 DynList (Class)

This interface is implemented by classes that wish to provide dynamic list capabilities. A dynamic list is a list of items that has one or more columns that can optionally be sorted, and the list can be filtered by column values or by global filters.

5.45.1.2.6 DynListDelegateSupporter (Class)

This interface contains functionality to support the DynListReqHdlrDelegate

5.45.1.2.7 DynListFilter (Class)

This interface is implemented by classes that are used to filter dynamic lists.

5.45.1.2.8 DynListSubject (Class)

This interface is implemented by classes that wish to be capable of being displayed in a dynamic list.

5.45.1.2.9 TSSDynList (Class)

This class implements the dynlist interface for TSS's in dynamic lists.

5.45.1.2.10TSSDynListSubject (Class)

This class implements the DynListSubject interface and contains fields for sorting and filtering TSS's in dynamic lists.

5.45.1.2.11TSSDynListSupporter (Class)

This class implements the DynListDelegateSupporter for creating dynamic lists containing TSS's

5.45.1.2.12TSSStatusFilter (Class)

This filter extends the DeviceStatusFilter to provide status filtering for TSS objects.

5.45.1.2.13WebObjectLocation Supporter (Class)

This interface allows common processing for objects supporting an ObjectLocation via the WebObjectLocation wrapper class..

5.45.1.2.14WebTSS (Class)

This class wraps the TransportationSystemSensor CORBA interface, caches data, and provides access to the cached data.

5.45.2 Sequence Diagrams

5.45.2.1 EditTSSLocationSupporter:setObjectLocation (Sequence Diagram)

This diagram shows the processing to save the TSS location when the user submits the Edit Location form. The SpecifyLocationReqHdlr calls the EditTSSLocationSupporter with the location parsed from the request. If the location is being edited while adding an RTMS, the configuration is retrieved and altered within the AddRTMSFormData object. If it is for an existing TSS, the WebTSS is called to get the TSS reference, the configuration is queried from the TSS, modified within the configuration, and the TSS is called to set the configuration. If successful, the cached configuration is updated with the altered configuration.

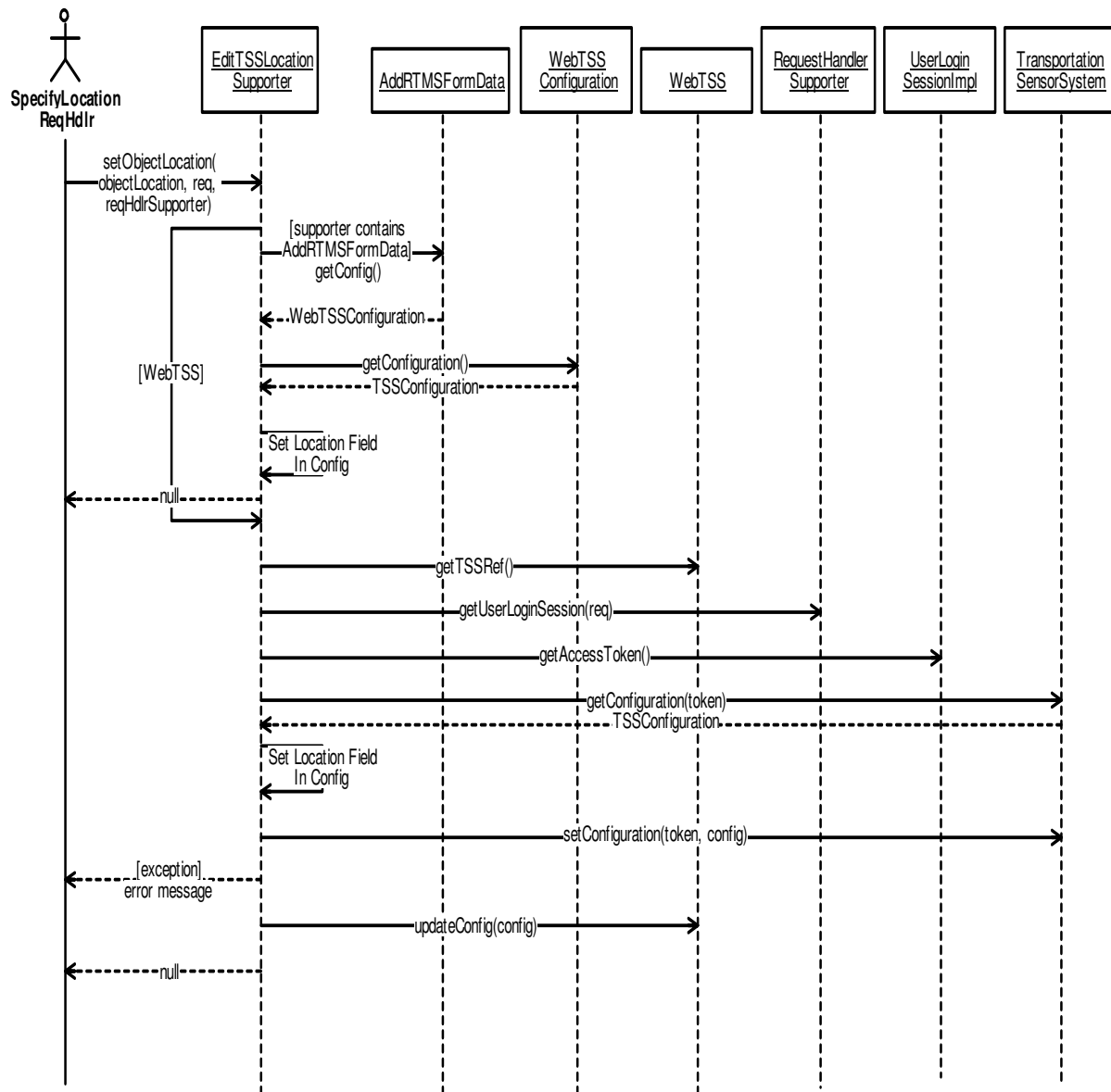


Figure 5-359. EditTSSLocationSupporter:setObjectLocation (Sequence Diagram)

5.45.2.2 TSSListSupporter:createDynList (Sequence Diagram)

This diagram shows how the Detector/TSS dynamic list is created for viewing on the Detector List page. A DefaultDynListCol object is created to represent each column. If a column has a filter, the filter object is created and set into the column. If the column is to be hidden by default, a call is made to the column set the default visibility. A new TSSDynList object is created using the columns that were created. If the request indicates that a global filter for the user's operation center folders should be used, the filter is created and added to the dynamic list. Other filter values include one of the device status values, which (if specified) is set in the device status filter. The setTSSTypeInclusionFlags() method is called to apply the external / internal TSS flags specified in the request.

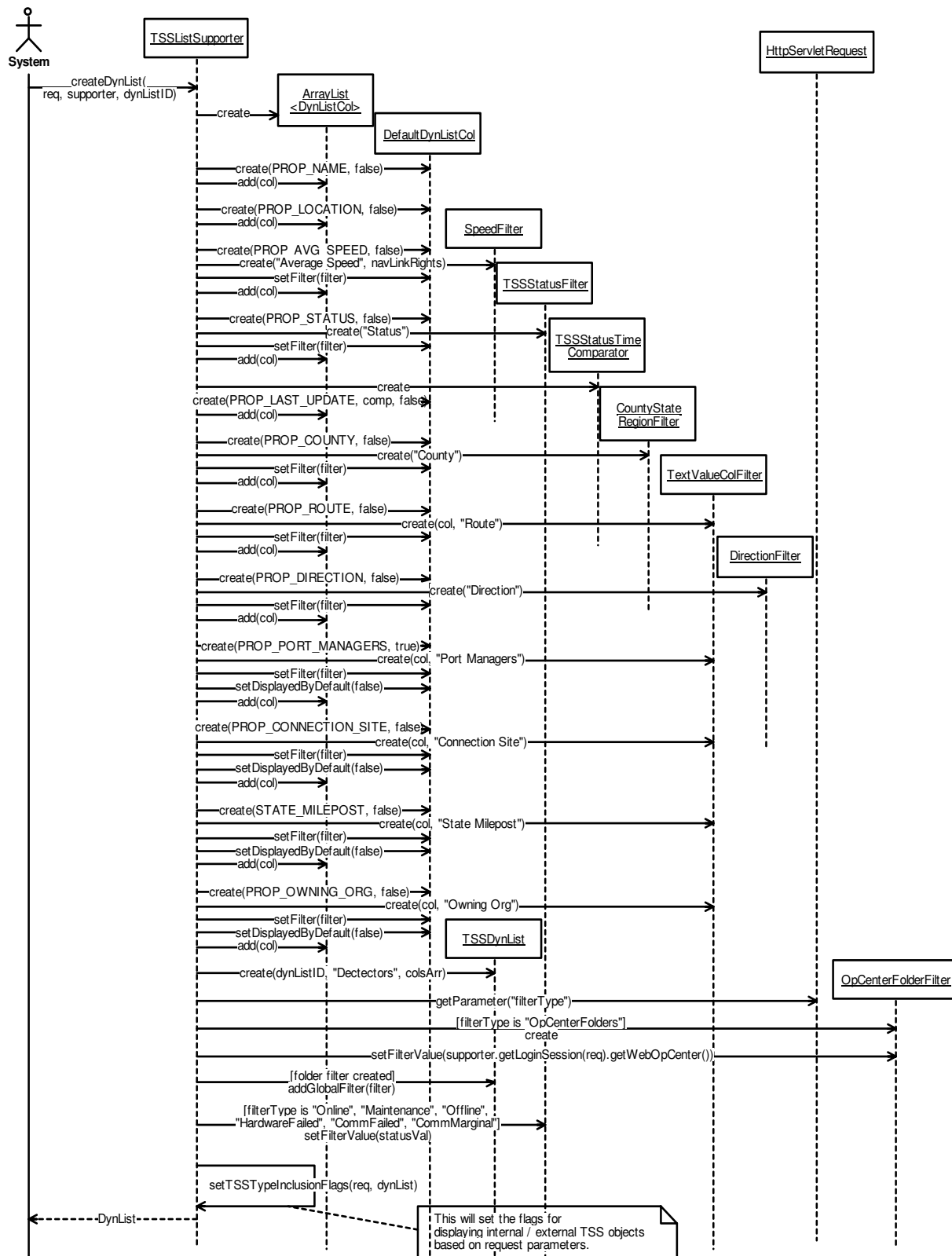


Figure 5-360. TSSListSupporter:createDynList (Sequence Diagram)

5.45.2.3 TSSReqHdlr:getEditTSSLocationForm (Sequence Diagram)

This diagram shows how the Edit TSS Location form is displayed. The formDataID and tssID parameters are parsed from the request, and one of these should be present (the formDataID if a TSS is being added or the tssID if editing the location of an existing TSS). If the formDataID is specified, the AddRTMSFormData object is retrieved from the TempObjectStore, the form fields are saved into the form data, and a new EditTSSLocationSupporter object is created. If the tss ID is specified, the WebTSS is retrieved from the cache, the user's rights are checked for the given TSS's organization, and the EditTSSLocationSupporter object is created. This object is added to the TempObjectStore and the response is redirected so that the displayEditObjectLocationDataForm request is invoked.

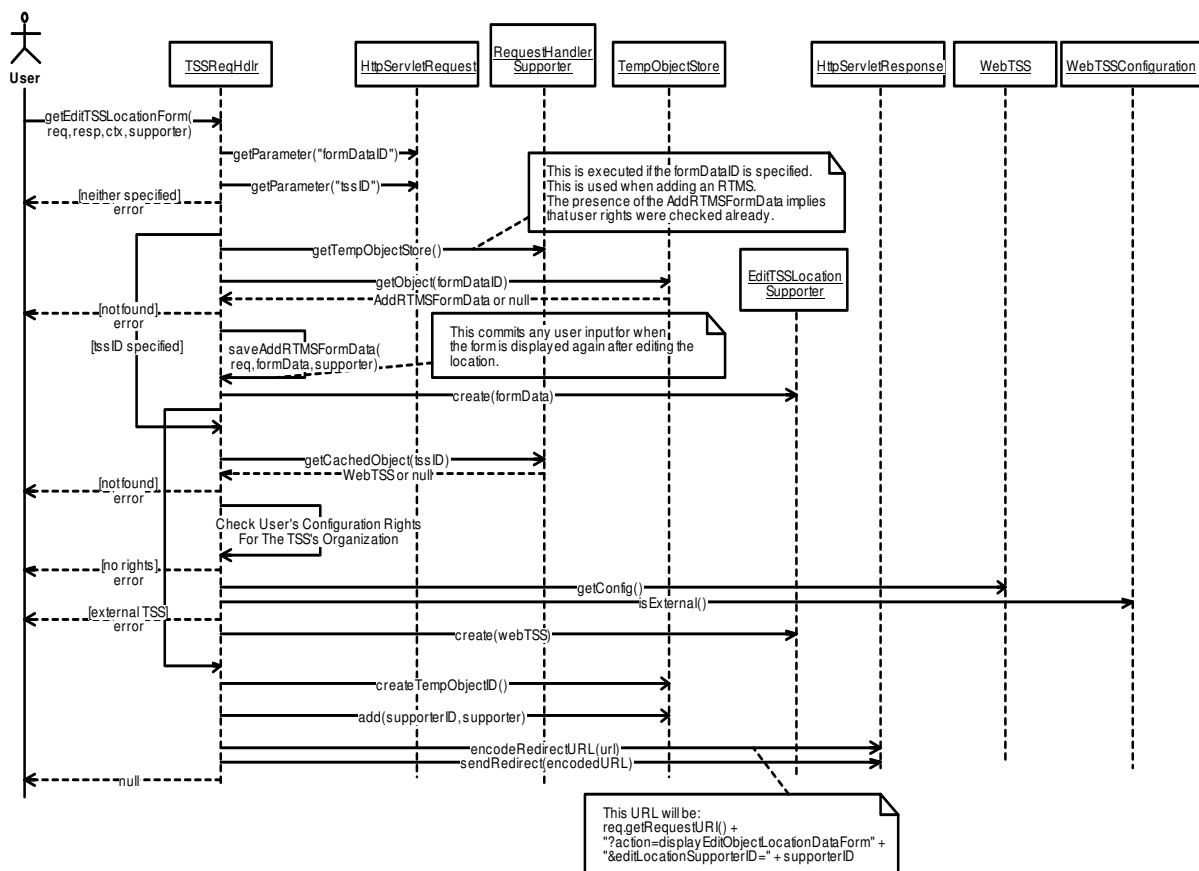


Figure 5-361. TSSReqHdlr:getEditTSSLocationForm (Sequence Diagram)

5.45.2.4 chartlite.servlet.tss:setTSSConfigCommSettings (Sequence Diagram)

This diagrams shows the processing that occurs when a TSS is configured for TCP/IP communications.

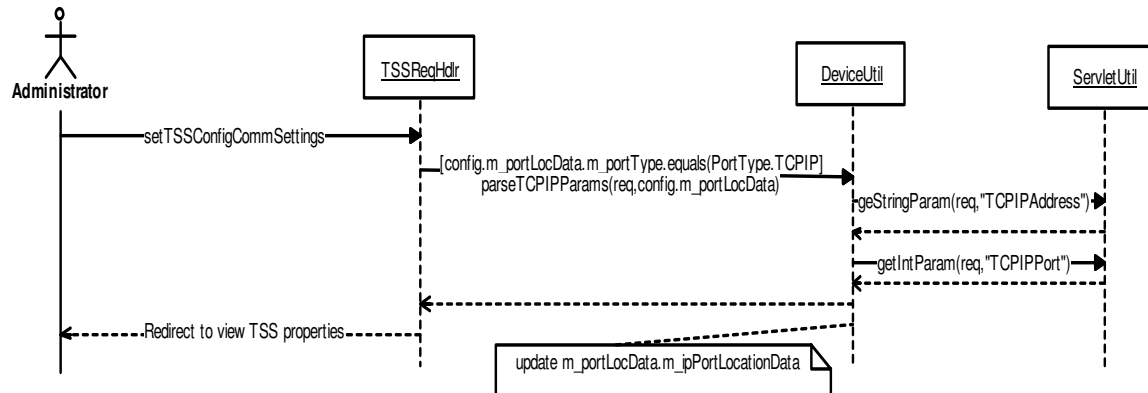


Figure 5-362. chartlite.servlet.tss:setTSSConfigCommSettings (Sequence Diagram)

5.46 Chartlite.servlet.servlet-dynlist

5.46.1 Classes

5.46.1.1 ServletDynListClasses (Class Diagram)

This diagram shows classes that support dynamic lists.

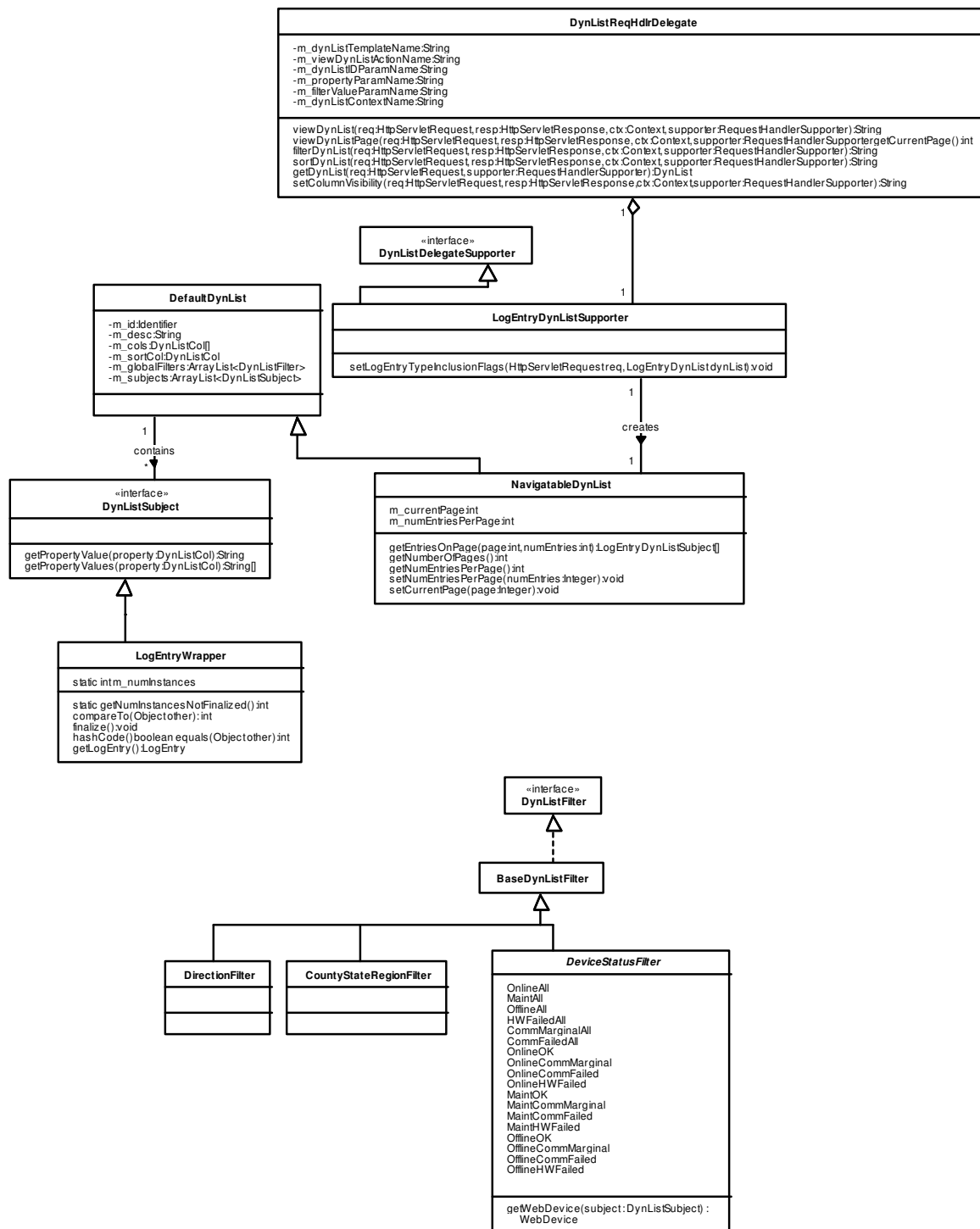


Figure 5-363. ServletDynListClasses (Class Diagram)

5.46.1.1.1 BaseDynListFilter (Class)

This abstract class provides a base implementation of the DynListFilter interface.

5.46.1.1.2 CountyStateRegionFilter (Class)

This class provides filtering on the county/state/region column of a dyn list. To use this filter, the objects that implement the DynListSubject interface must also implement the WebObjectLocationSupporter interface to be able to supply a WebObjectLocation for the subject.

5.46.1.1.3 DefaultDynList (Class)

This class provides a default implementation of the DynList interface. It supports a collection of columns, a collection of global filters, and a collection of subjects. Filters in this list are treated additively - that is, a subject must pass all filters to be displayed.

5.46.1.1.4 DeviceStatusFilter (Class)

This class is a base filter for filtering on the device status. It must be extended to get the WebDevice from the DynListSubject object.

5.46.1.1.5 DirectionFilter (Class)

This class provides filtering on the direction column of a dyn list. To use this filter, the objects that implement the DynListSubject interface must also implement the WebObjectLocationSupporter interface to be able to supply a WebObjectLocation for the subject.

5.46.1.1.6 DynListDelegateSupporter (Class)

This interface contains functionality to support the DynListReqHdlrDelegate

5.46.1.1.7 DynListFilter (Class)

This interface is implemented by classes that are used to filter dynamic lists.

5.46.1.1.8 DynListReqHdlrDelegate (Class)

This class helps request handlers support dynamic lists. Requests to view, sort, or filter dynamic lists can be passed from a request handler to this class, provided the URL used for the requests contain parameters required by this class, such as the id of the list, the property name, and/or the filter value.

5.46.1.1.9 DynListSubject (Class)

This interface is implemented by classes that wish to be capable of being displayed in a dynamic list.

5.46.1.1.10 LogEntryDynListSupporter (Class)

This class is a DynListDelegateSupporter for comm log entries. Its implementation of the createDynList method sets up the columns for the list of comm log entries. Its implementation of the getDynListSubjects method retrieves the log entries from the GUI's

CommLogManager and wraps them as LogEntryDynListSubjects.

5.46.1.1.11 LogEntryWrapper (Class)

This class provides accessor methods that provide access to the information in a CHART2.LogCommon.LogEntry object. In R3B2 this class implements the DynListSubject interface for use in DynLists

5.46.1.1.12 NavigatableDynList (Class)

This object extends the DefaultDynList class to provide page navigation of DynListSubjects

5.46.2 Sequence Diagrams

5.46.2.1 chartlite.servlet.dynlist.DynListReqHdlrDelegate:createDynList (Sequence Diagram)

This diagram shows the processing that occurs when a dynamic list is created. Starting in R3B3, the operator may update the visible columns in the list. The visible columns data is stored in a cookie, on a per device type basis. If a cookie is found, display settings are set from it. If no cookie exists, a new one is created using the default column display values.

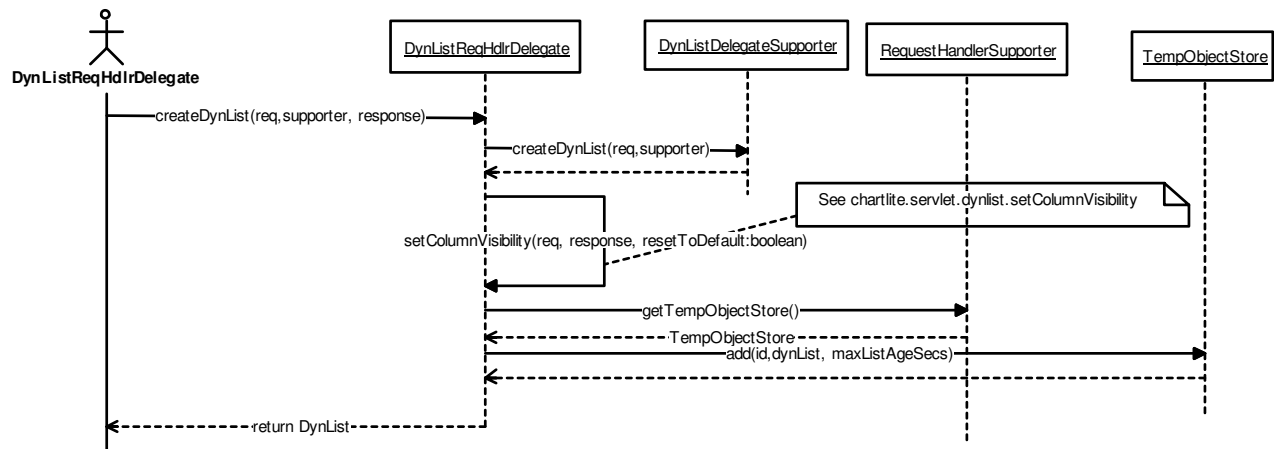


Figure 5-364. chartlite.servlet.dynlist.DynListReqHdlrDelegate:createDynList (Sequence Diagram)

5.46.2.2 chartlite.servlet.dynlist.DynListReqHdrDelegate:setColumnVisibility (Sequence Diagram)

This diagram shows the processing that occurs for retrieving and setting the cookie containing which columns to display in a dyn list. This method is called after initial creation of a dynlist or when a request comes in to update the column settings. When an operator updates the column settings, the cookie is updated using javascript.

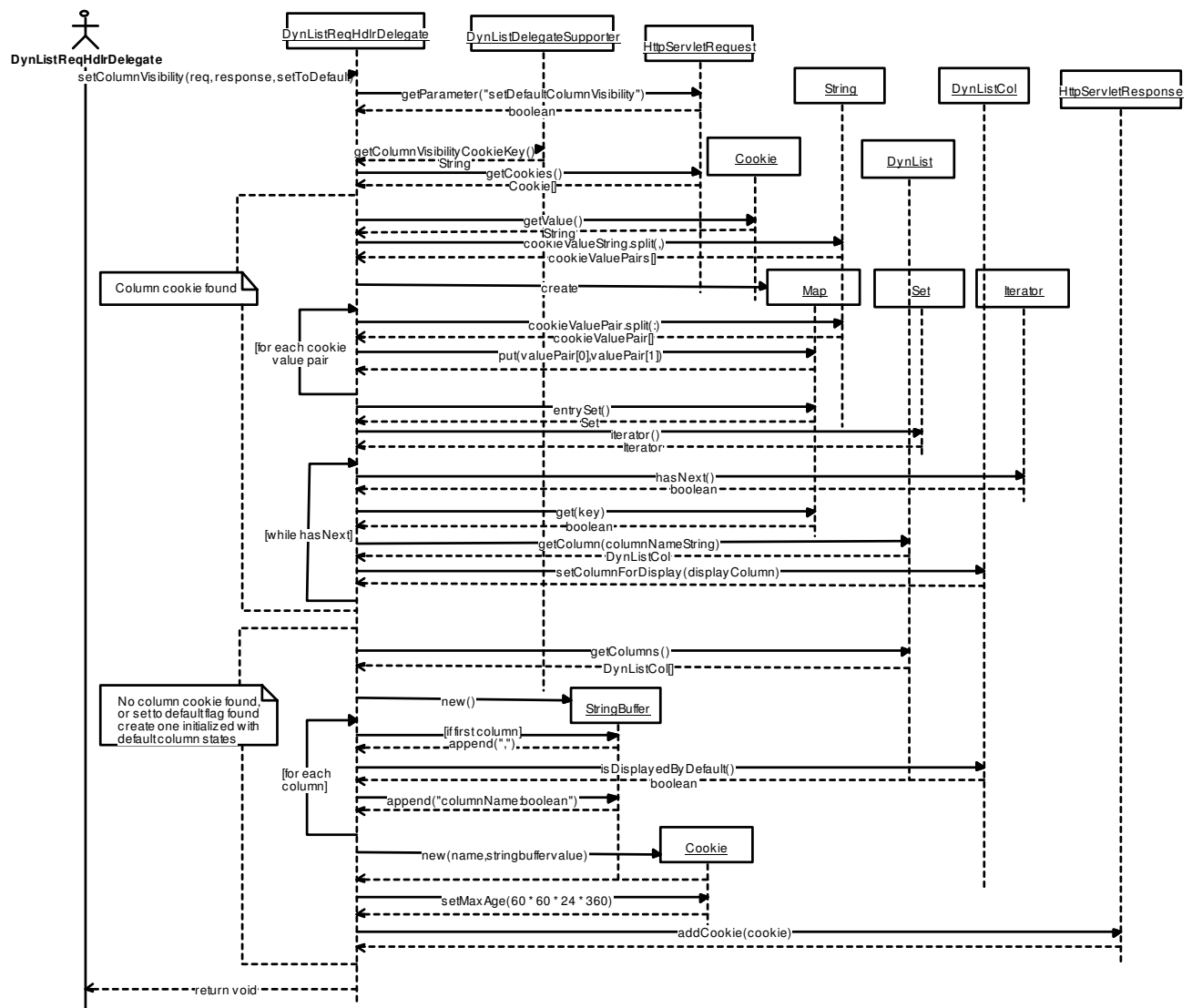


Figure 5-365. chartlite.servlet.dynlist.DynListReqHdrDelegate:setColumnVisibility (Sequence Diagram)

5.47 Chartlite.servlet.dms

5.47.1 Class diagrams

5.47.1.1 chartlite.servlet.dms.dynlist_classes (Class Diagram)

This diagram shows class items that extend the existing dms list functionality to include new filter columns and the ability to hide/show internal and external DMS'.

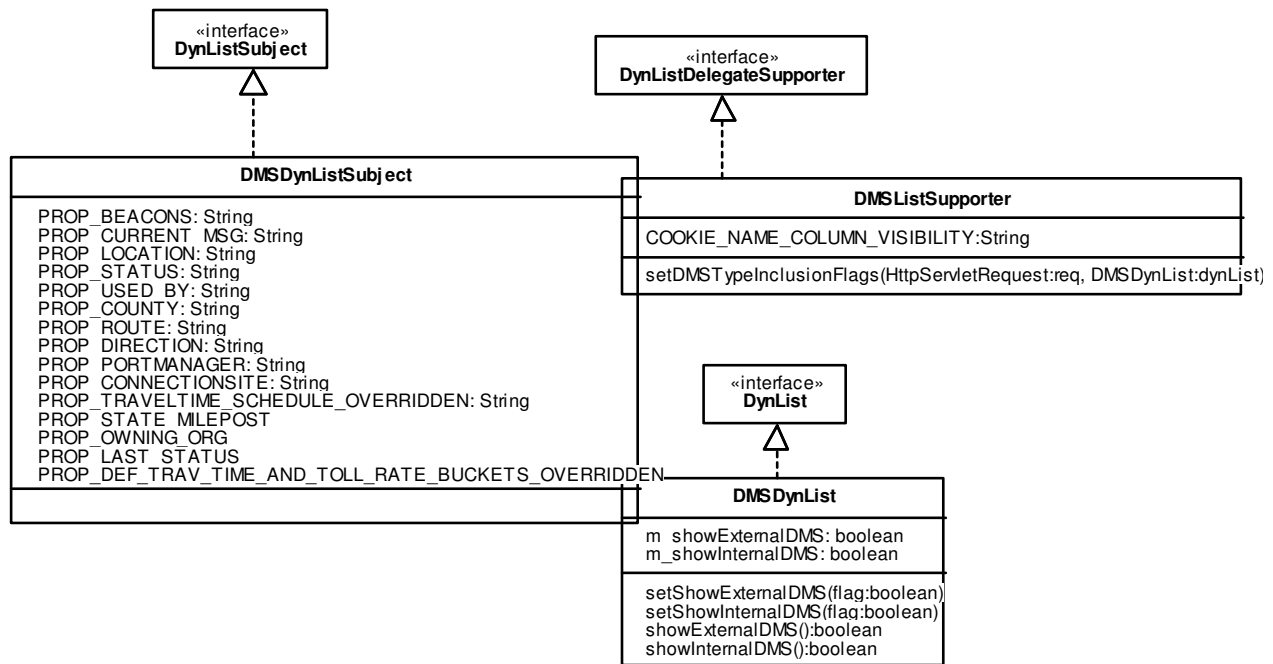


Figure 5-366. chartlite.servlet.dms.dynlist_classes (Class Diagram)

5.47.1.1.1 DMSDynList (Class)

This class implements the dynlist interface for external devices in dynamic lists.

5.47.1.1.2 DMSDynListSubject (Class)

This class implements the DynListSubject interface and contains fields for displaying DMS's in dynamic lists.

5.47.1.1.3 DMSListSupporter (Class)

This class implements the DynListDelegateSupporter for creating dynamic lists of DMS's

5.47.1.1.4 DynList (Class)

This interface is implemented by classes that wish to provide dynamic list capabilities. A dynamic list is a list of items that has one or more columns that can optionally be sorted, and the list can be filtered by column values or by global filters.

5.47.1.1.5 DynListDelegateSupporter (Class)

This interface contains functionality to support the DynListReqHdlrDelegate

5.47.1.1.6 DynListSubject (Class)

This interface is implemented by classes that wish to be capable of being displayed in a dynamic list.

5.47.1.2 GUIDMSServletClasses (Class Diagram)

This diagram shows CHART GUI servlet classes related to dynamic message signs.

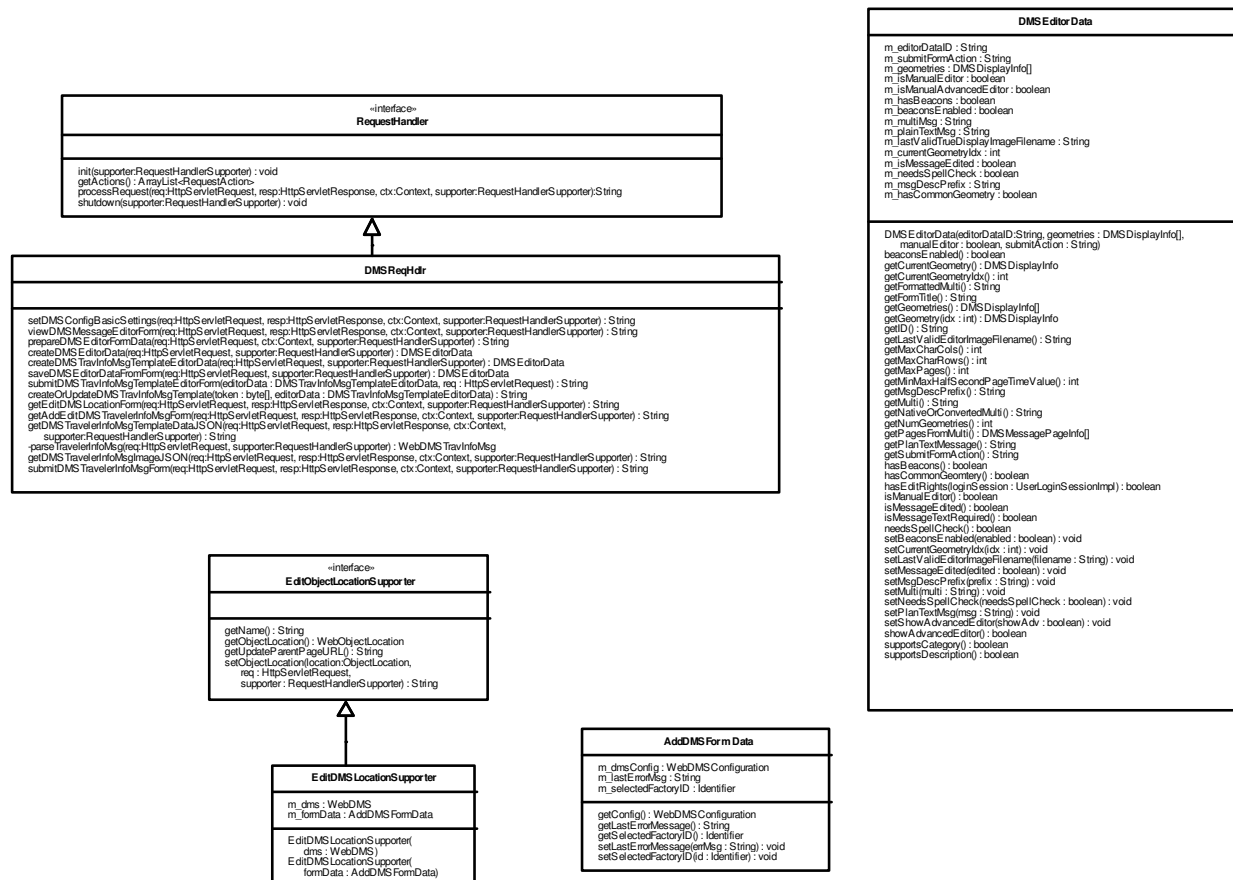


Figure 5-367. GUIDMSServletClasses (Class Diagram)

5.47.1.2.1 AddDMSFormData (Class)

This class represents the data in the Add DMS and Copy DMS forms.

5.47.1.2.2 DMSEditorData (Class)

This class represents an instance of a DMS message being edited in an editor. It provides storage so that the message and editor state can be preserved during interim requests before the form is submitted. It also has logic for manipulating the editor session. This is a base class and will be extended for specific editor types.

5.47.1.2.3 DMSReqHdlr (Class)

This class is a request handler used to process requests related to dynamic message signs (DMS).

5.47.1.2.4 EditDMSLocationSupporter (Class)

This class is used to support editing the location of an existing or new DMS.

5.47.1.2.5 EditObjectLocationSupporter (Class)

This interface provides functionality allowing the location data to be edited. (For example, the target of the edited location may be an existing object, or it may be a form data object for creating a new object).

5.47.1.2.6 RequestHandler (Class)

This interface specifies methods that are to be implemented by classes that are used to process requests

5.47.2 Sequence Diagrams

5.47.2.1 `chartlite.servlet.dms:createDMSEditorData` (Sequence Diagram)

This diagram shows the creation of the new type of DMS editor data to support the new DMS message templates. If the "editTravInfoMsgTemplate" parameter is present, a call will be made to `createDMSTravInfoMsgTemplateData()`. If the user is editing an existing template, the "templateID" parameter will be passed and this ID will be used to retrieve the cached template wrapper object from the template factory wrapper, and the rows and columns will be queried from the template. If the templateID is not specified, the user is editing a new template, and the rows and columns must be specified in the request to specify the template's size. The template factory wrapper is called to retrieve the list of all formats for each appropriate type of tag. A new `DMSTravInfoMsgTemplateEditorData` object is created to represent the editor state. If editing an existing template, the formats used in the template will be added to the list of available formats if necessary so that they are guaranteed to be in the selectable lists. The flag to show the advanced version of the editor will be set to true. If `createDMSEditorData()` is invoked for other (existing) DMS editor types, the functionality will remain unchanged.

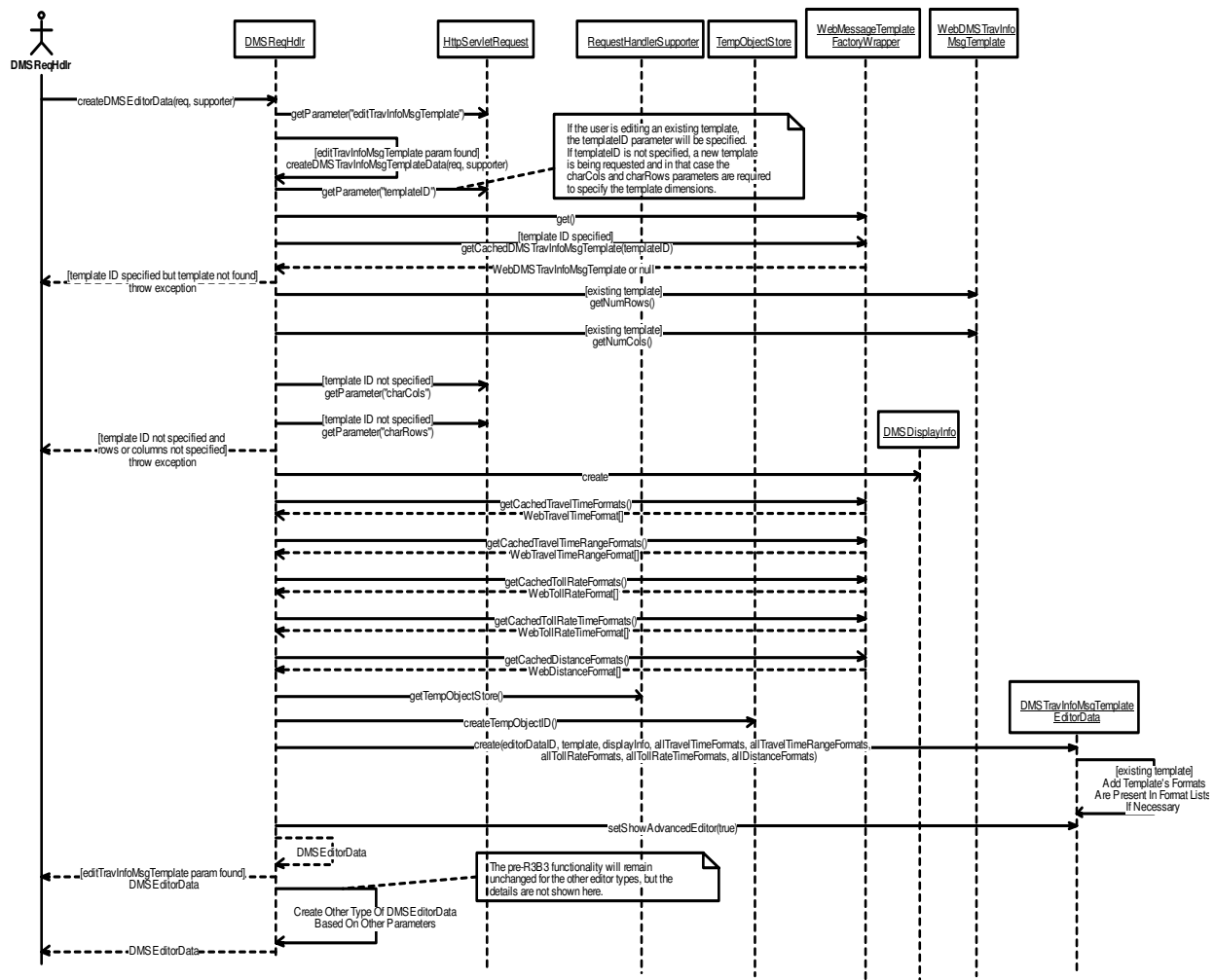


Figure 5-368. chartlite.servlet.dms:createDMSEditorData (Sequence Diagram)

5.47.2.2 chartlite.servlet.dms:createOrUpdateDMSTravInfoMsgTemplate (Sequence Diagram)

This diagram shows the processing to create or update a DMS message template. It is called after the editor data has been checked and determined to be OK to submit. The template configuration and the WebDMSTravInfoMsgTemplate object are retrieved from the editor data. A null template indicates that the template is being added, while a non-null template indicates an edit operation. The template factory wrapper is called to check whether the description is a duplicate of an existing template (excluding the current template if it's an edit operation) and an error is returned if it is a duplicate. If creating a new template, the template factory wrapper is called, and it iterates through the factories that are known to the system. If the template was created, a new WebDMSTravInfoMsgTemplate wrapper object is created and added to the cached templates. If it is an edit operation, the template's CORBA reference is retrieved and called to set the configuration, after which the cached configuration is updated within the existing WebDMSTravInfoMsgTemplate wrapper object.

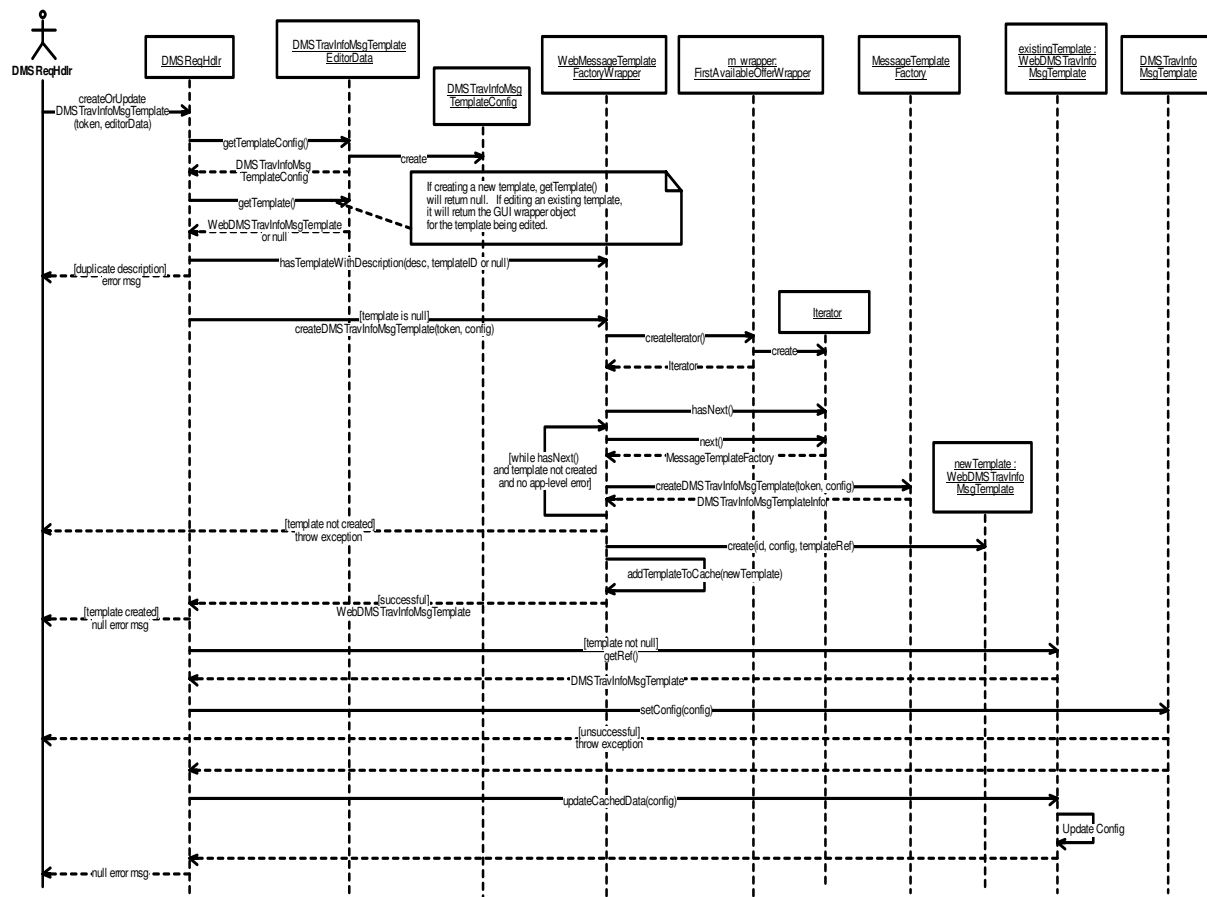


Figure 5-369. chartlite.servlet.dms:createOrUpdateDMSTravInfoMsgTemplate (Sequence Diagram)

5.47.2.3 chartlite.servlet.dms:getAddEditDMSTravInfoMsgForm (Sequence Diagram)

This diagram shows how the form is displayed for adding or editing a DMS traveler info message. The DMS ID is specified in the request and is used to look up the WebChart2DMS object from the cache. After checking for the necessary user rights, the DMS's character columns and rows are obtained and all cached WebDMSTravInfoMsgTemplate objects are checked against these dimensions. If the rows and columns are the same as that of the DMS, and if the number of pages does not exceed the DMS's max pages, the template is added to a list. If it is an edit operation, the travInfoMsgID parameter will be specified and will be used to look up the cached WebDMSTravInfoMsg object. The templates, message, and DMS wrapper objects are put into Velocity context so that the Velocity can access the data while building the HTML to return.

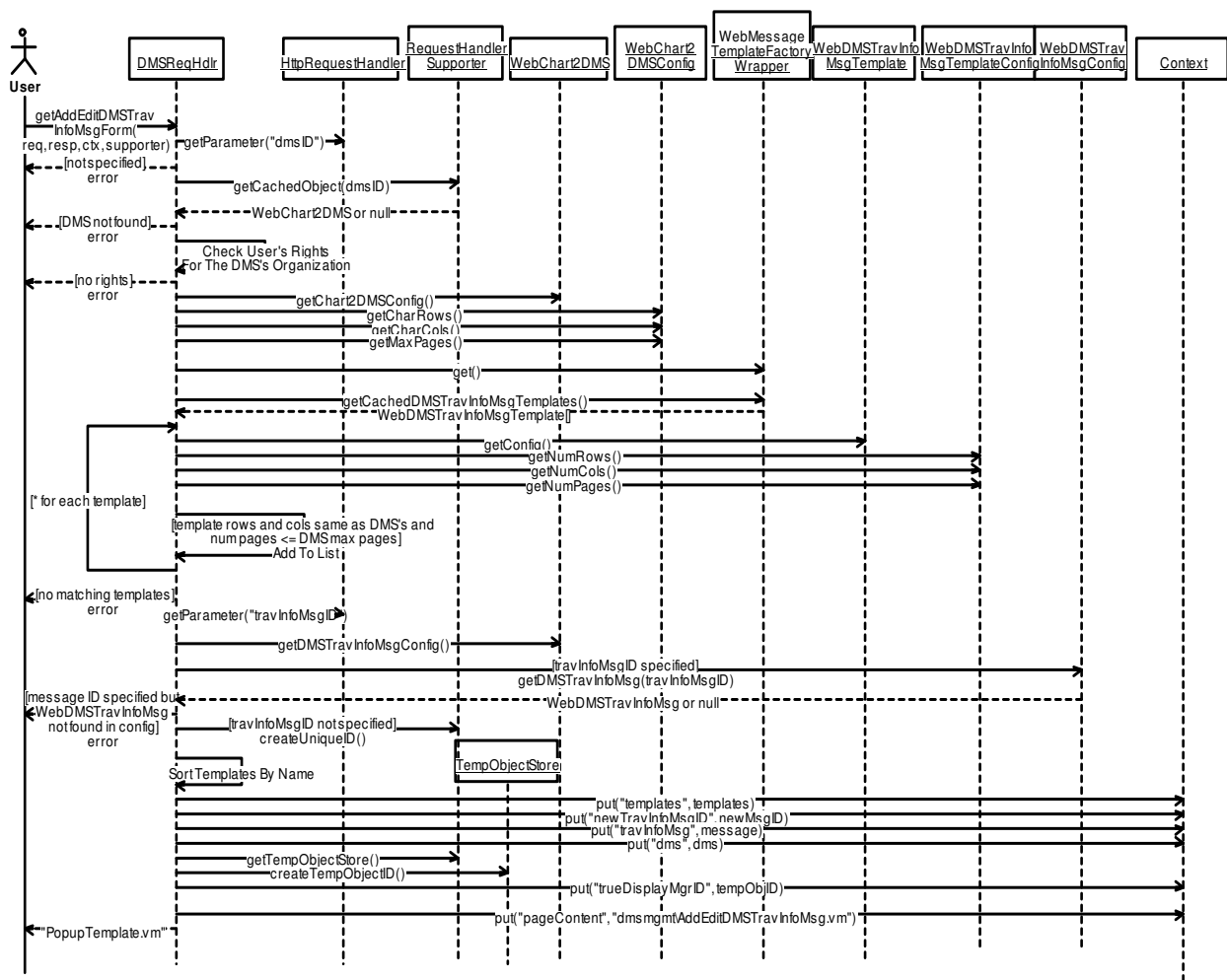


Figure 5-370. chartlite.servlet.dms:getAddEditDMSTravInfoMsgForm (Sequence Diagram)

5.47.2.4 chartlite.servlet.dms:getDMSEditorImageJSON (Sequence Diagram)

This diagram shows how the DMS editor image will be generated. First the current form inputs are saved into the editor data. The editor may represent more than one sign size (or "geometry") but the image is only displayed for one size. The current geometry index is used unless otherwise specified by the "selectedGeometryIdx" parameter. The formatted MULTI is queried from the editor data. For the template editor, this will cause the template tags to be substituted with dummy data. For other manual editors, the MUTLI will be returned as is, while for automatic formatting editors, the returned MULTI will be null. If MULTI is returned, its "message statistics" are counted (i.e., the number of rows, columns, and pages required for the message). The list of geometries represented by the editor is checked against the MULTI stats to find the bad geometry indices (the sign sizes too small to display the message), and for the selected index, flags for too many rows/columns/pages are calculated. If the message is plain text (from an auto editor), the plain text is converted to MULTI for each geometry and the bad geometry and too many rows/pages flags are set on failure. The selected geometry index is used, and the plain text is converted to MULTI, and the GIF image is created. If successful, the filename is set into the editor data, and the selected geometry index is set as the current index in the editor data. Finally a JSON response object is created that contains the filename, current geometry index, image dimensions, and error information, and the JSON object is sent back to the browser. When the browser receives the object, it will be able to use the fields using Javascript.

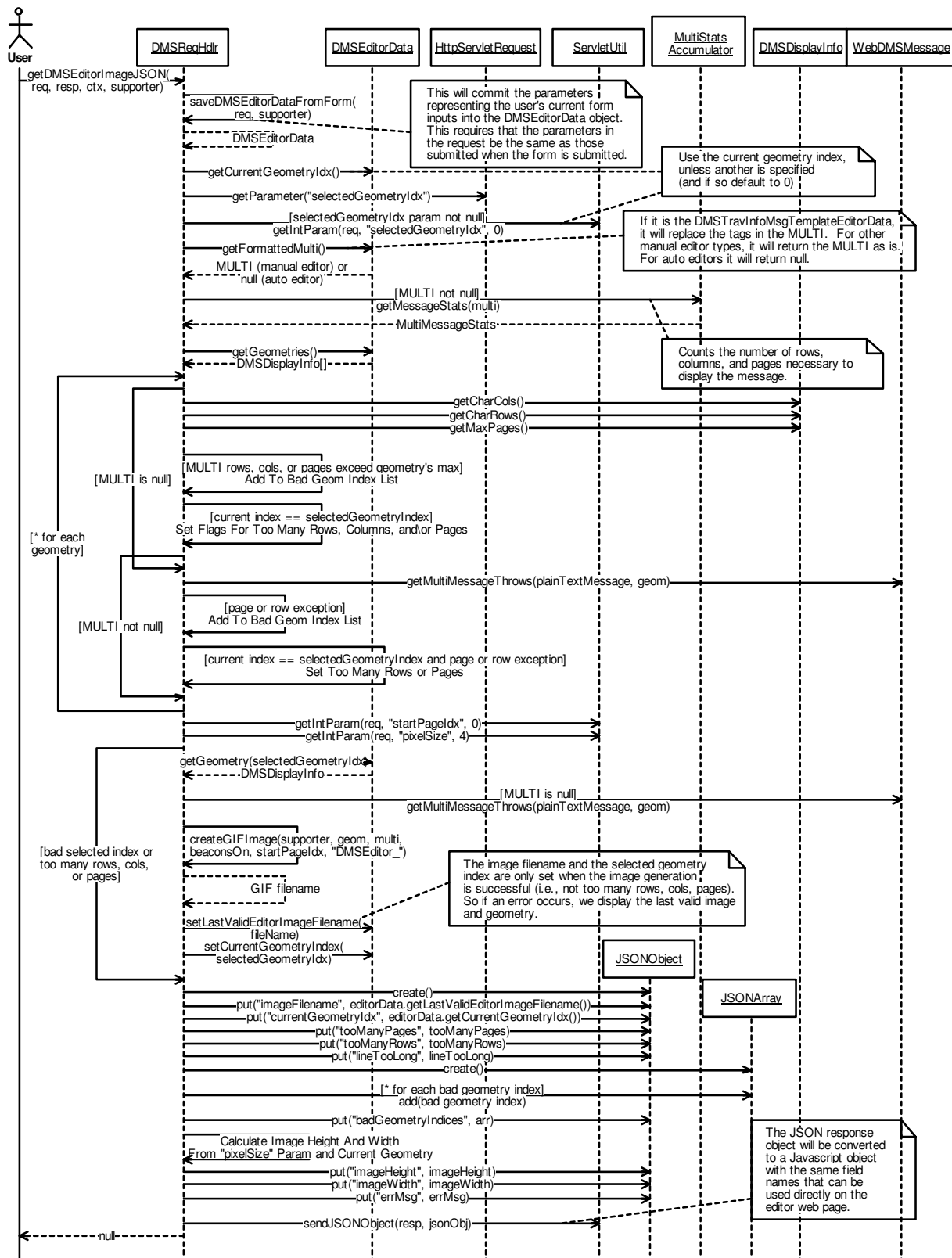


Figure 5-371. chartlite.servlet.dms:getDMSEditorImageJSON (Sequence Diagram)

5.47.2.5 chartlite.servlet.dms:getDMSTravInfoMsgImageJSON (Sequence Diagram)

This diagram shows how the true display image is generated for the Add / Edit DMS Traveler Info Message Form. The parameters are parsed and a new DMSTravInfoMsg structure is created, as shown in the parseDMSTravInfoMsg sequence diagram. The DMS ID is used to retrieve the WebDMS object from the cache. The specified true display manager ID is used to look up the DMSTravInfoMsgTrueDisplayMgr from the TempObjectStore, which may have been put there by a previous invocation of this method for the same editor session. If it is found, the DMSTravInfoMsg is updated within the true display manager. If not found, new WebDMSTravInfoMsg and DMSTravInfoMsgTrueDisplayMgr objects are created, and the latter is added to the TempObjectStore for future use. The manager is called to update its current GIF image, as shown in the DMSTravInfoMsgTrueDisplayMgr:updateGIF sequence diagram. The GIF filename and image dimensions (or error conditions) are then put into a new JSON object, which is sent back to the browser in the response. The JSON object will be interpreted by a Javascript Ajax response handler that will update the image on the page.

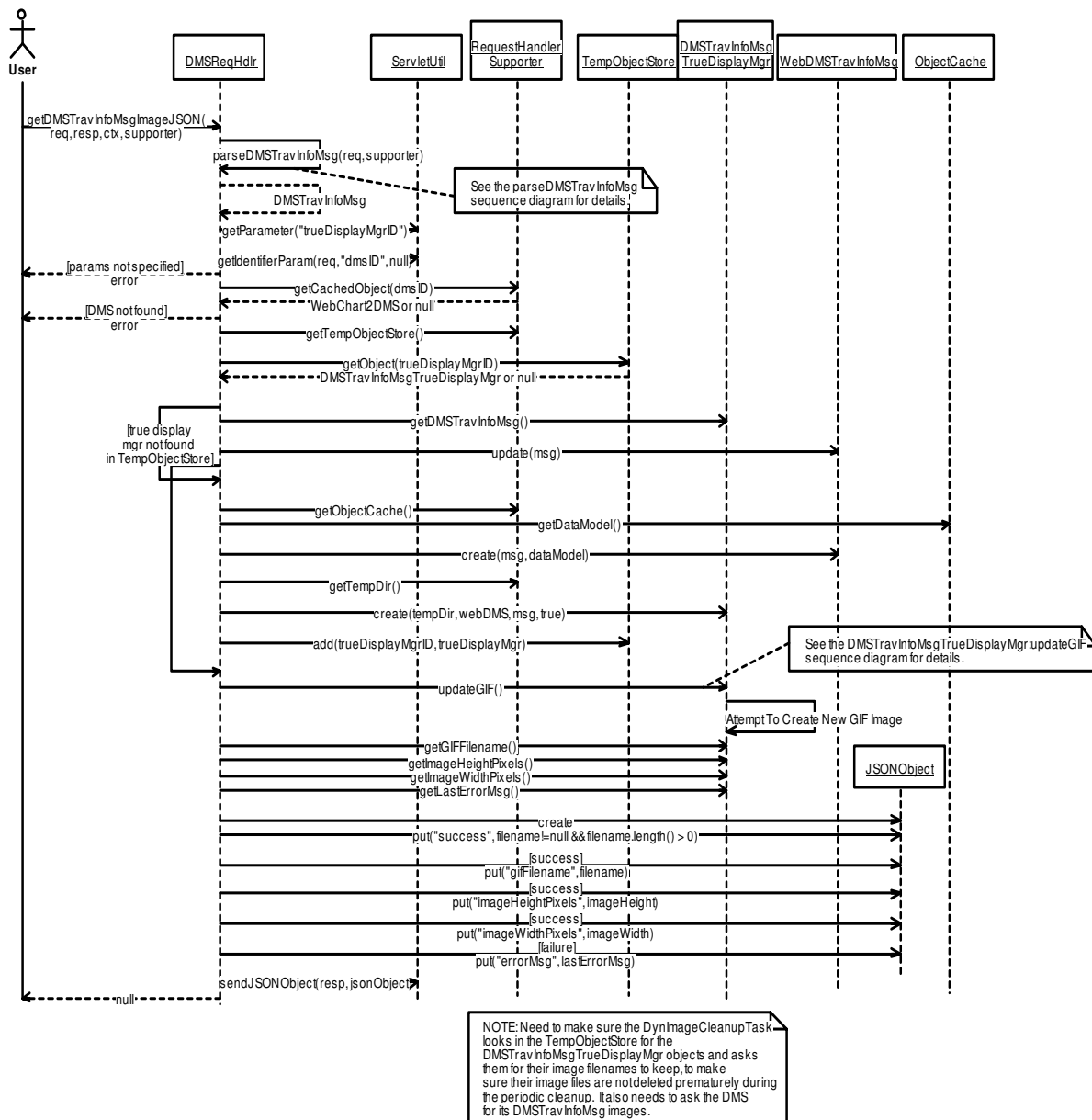


Figure 5-372. chartlite.servlet.dms:getDMSTravInfoMsgImageJSON (Sequence Diagram)

5.47.2.6 chartlite.servlet.dms:getDMSTravInfoMsgTemplateDataJSON (Sequence Diagram)

This diagram shows the processing when a user selects a template in the Add/Edit DMS Traveler Info Message form. The DMS and template IDs are used to retrieve the cached wrapper objects, and the route IDs associated with the DMS are used to retrieve the cached WebTravelRoute objects. This is a superset of the routes that can be applicable for each route index used in the template. Next the template is parsed to build a model, to determine which data elements are required for each route index used in the template. For each route index used, any routes that support the required travel time / toll rate data are added to the list. Finally a JSON response object is built containing the applicable routes for each route in the template, the template message text, and the template ID. The JSON response object is then sent back to the browser, which will parse the response into a Javascript object to be used by the Ajax response handler.

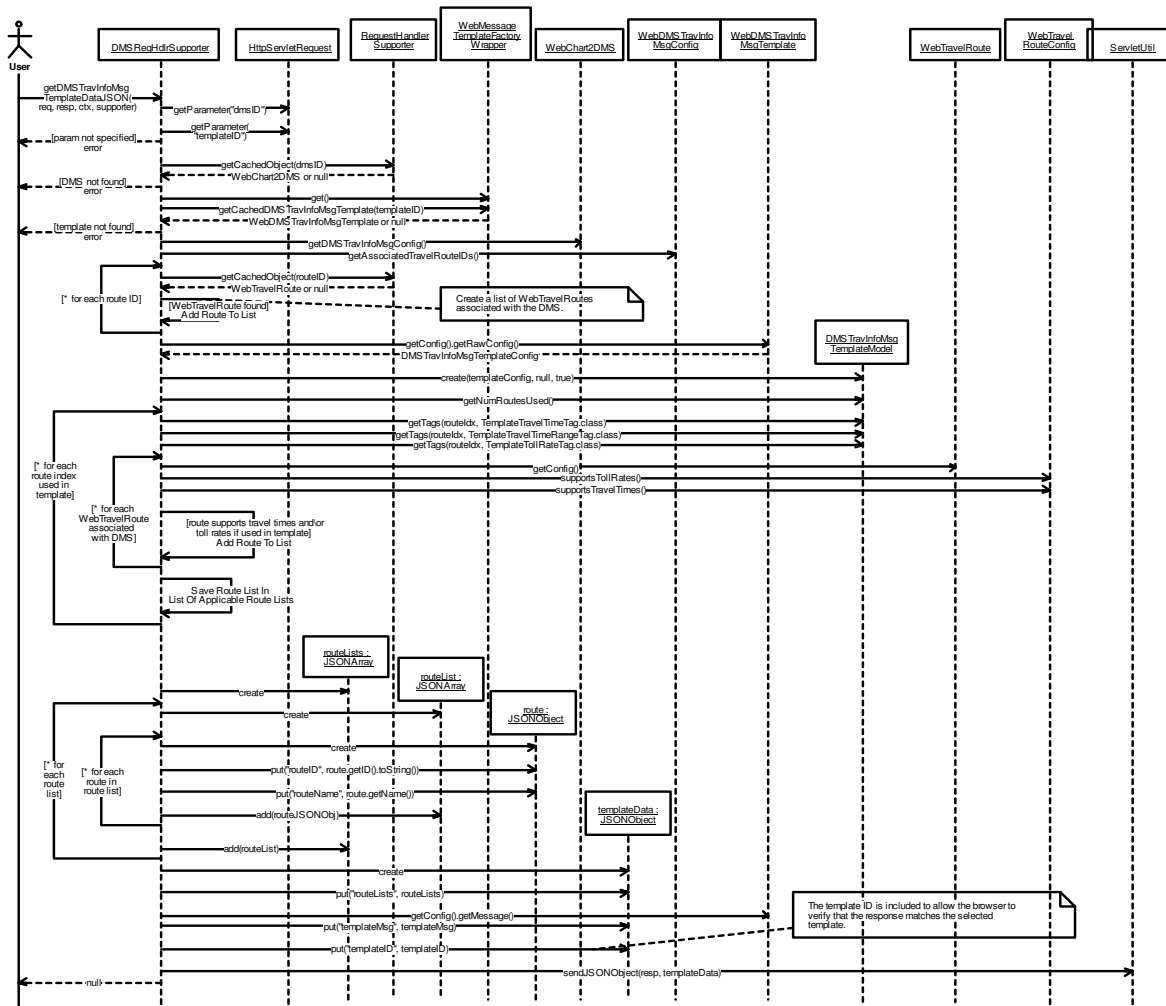


Figure 5-373. chartlite.servlet.dms:getDMSTravInfoMsgTemplateDataJSON (Sequence Diagram)

5.47.2.7 chartlite.servlet.dms:parseBasicConfigSettings (Sequence Diagram)

This diagram shows the processing done by the DMSReqHdlr to parse the parameters passed from the DMS basic configuration data form. This existing method is being modified for R3B1 to allow the responsible operations center to be set. This optional field will be used to specify which operations center should receive a device failure alert for a DMS. When not set, device failure alerts for the DMS are disabled. This method is modified for R3B3 to include NTCIP font selection and spacing. Other new fields for R3B3 include device and comm failure notification groups in addition to travel time and toll rate arb queue buckets.

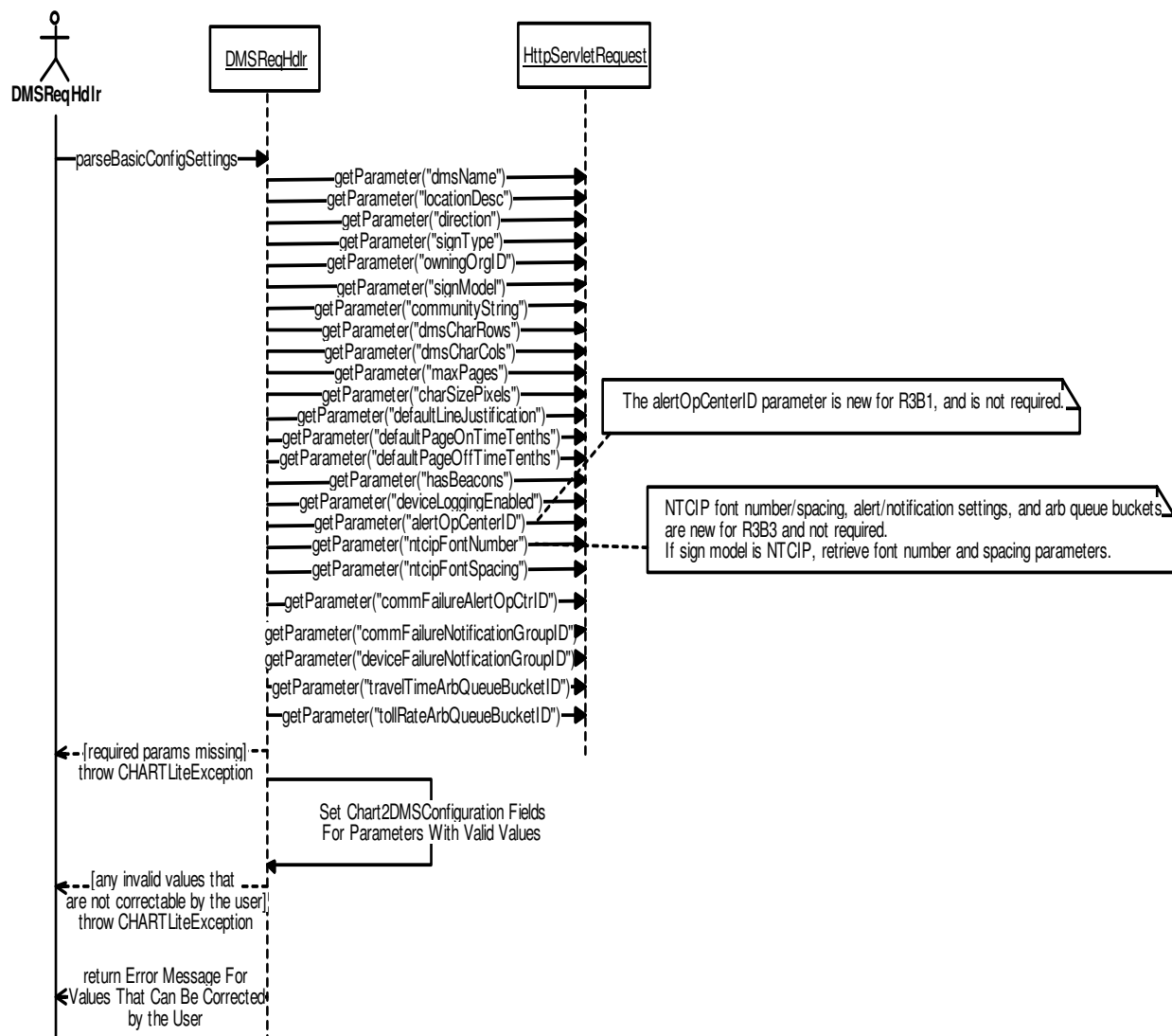


Figure 5-374. chartlite.servlet.dms:parseBasicConfigSettings (Sequence Diagram)

5.47.2.8 chartlite.servlet.dms:parseDMSTravInfoMsg (Sequence Diagram)

This diagram shows how a DMSTravInfoMsg structure is parsed from the form parameters when editing a DMS Traveler Info Message. This will be used when submitting the form and when updating the true display image for the editor.

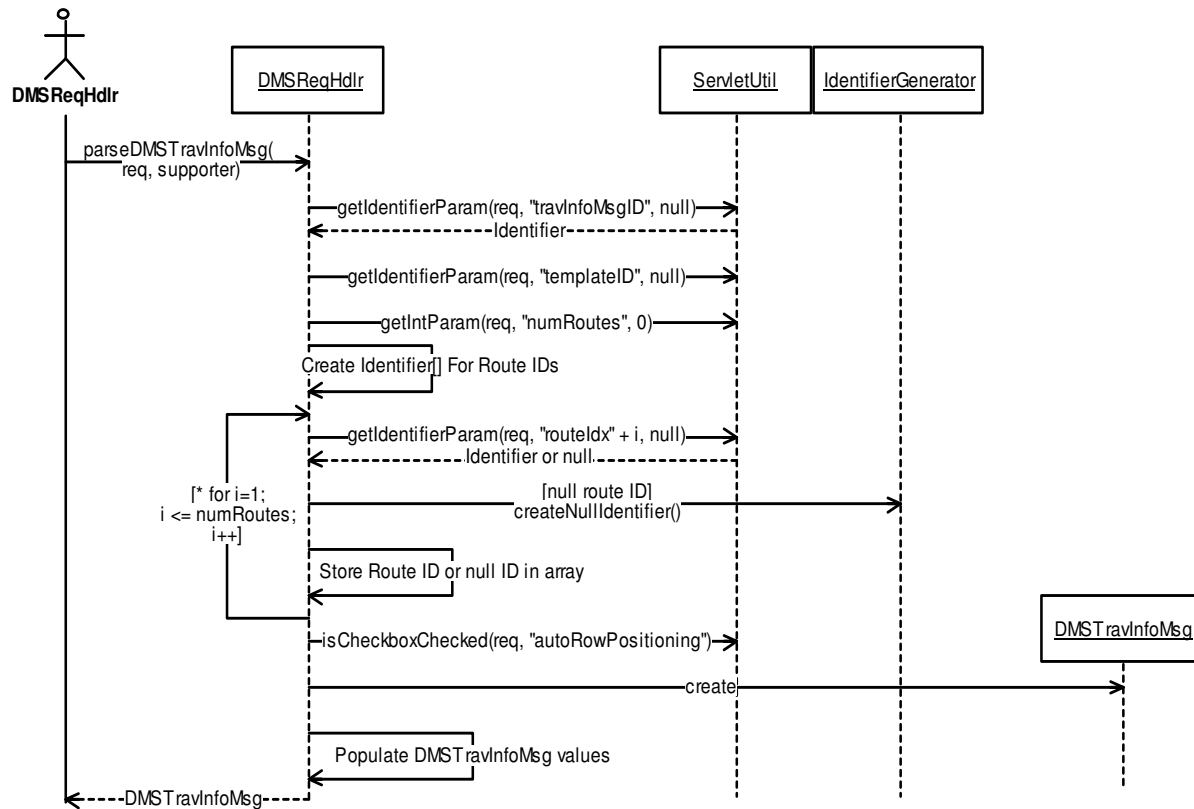


Figure 5-375. chartlite.servlet.dms:parseDMSTravInfoMsg (Sequence Diagram)

5.47.2.9 chartlite.servlet.dms:removeDMSTravInfoMsg (Sequence Diagram)

This diagram shows how a DMS traveler information message is removed from a DMS. The dmsID parameter is used to retrieve the WebChart2DMS from the cache. After checking the user rights, a copy of the latest configuration is queried from the Chart2DMS CORBA object in the server. The request handler builds a list, adding any DMSTravInfoMsg objects except for the one that matches the message ID. The messages are put back into the copy of the configuration, and the Chart2DMS is called again to set the configuration. If successful, the message is removed from the cached configuration and the WebDMSTravInfoMsg is removed. The response is redirected to display the DMS Details page to show the updated DMS configuration.

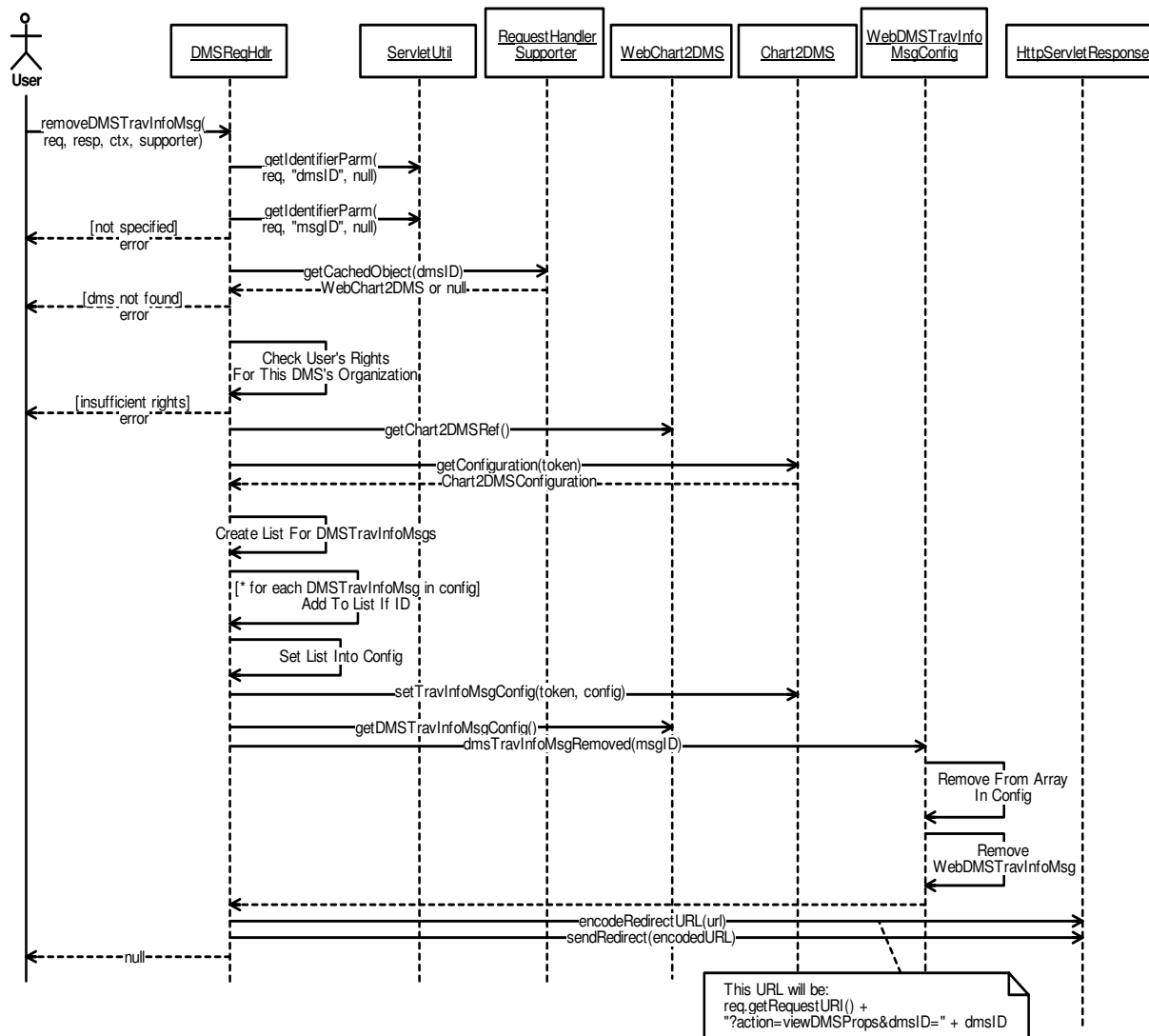


Figure 5-376. chartlite.servlet.dms:removeDMSTravInfoMsg (Sequence Diagram)

5.47.2.10 chartlite.servlet.dms:saveDMSEditorDataFromForm (Sequence Diagram)

This diagram shows how data is saved from the DMS message editor form into the DMSEditorData object corresponding to the editor type. The editor data ID is used to look up the DMSEditorData object that was previously stored in the TempObjectStore. If found, and the formDataSaved parameter is specified, it will just return the editor data without further processing. Otherwise, the parameters are read from the request and set into the editor data including the beacon state, message edited flag, advanced editor flag, the MULTI message (for the manual editor) or plain text message (for the auto editor). If the editor data type indicates the template editor, the type-specific parameters are parsed and stored in the editor data including the template description, template field formats, destination tag alignment, and policy to follow if the route data is missing. For the RPI or Stored Message editors, the description and/or category are parsed and stored in the editor data.

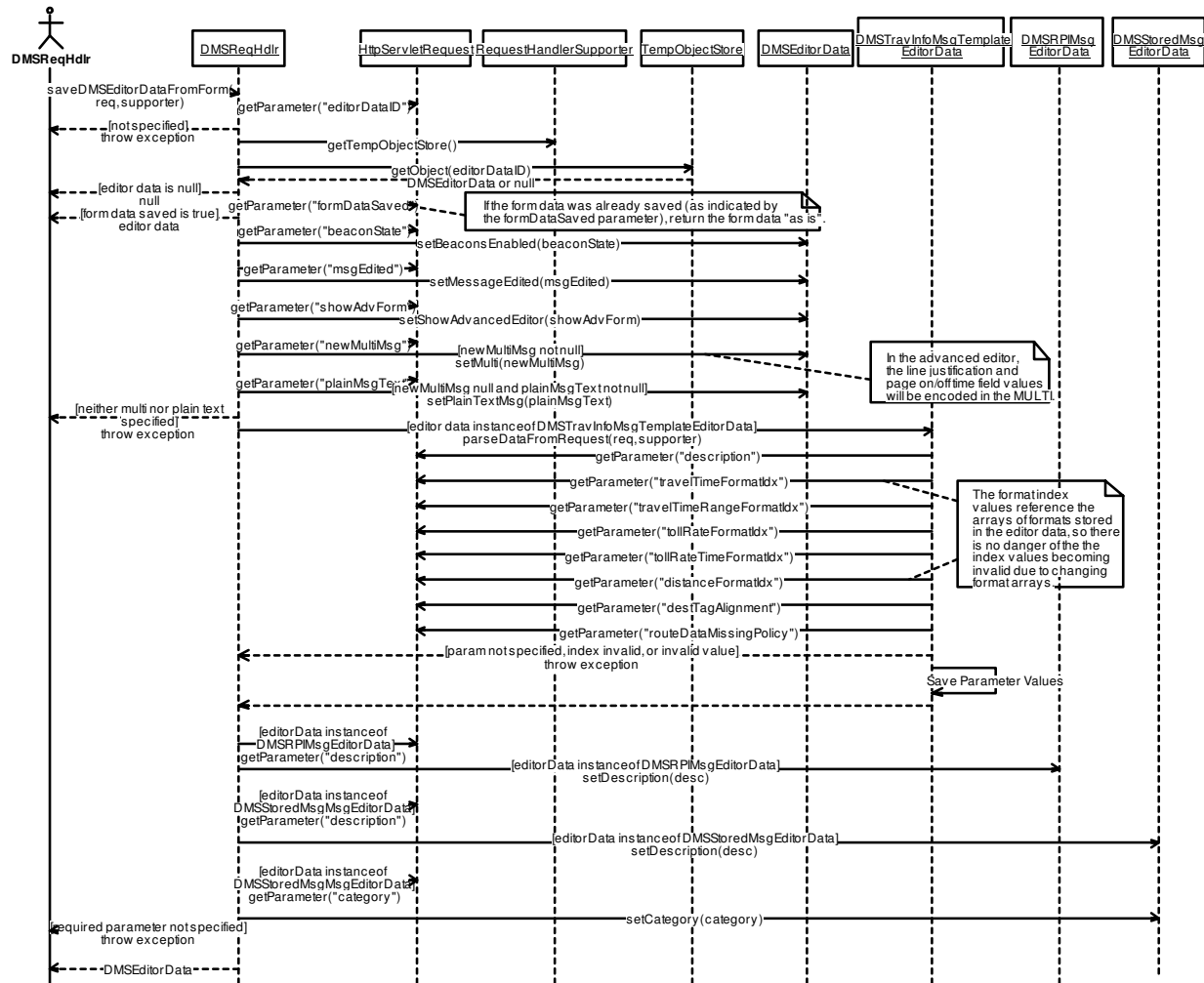


Figure 5-377. chartlite.servlet.dms:saveDMSEditorDataFromForm (Sequence Diagram)

5.47.2.11 chartlite.servlet.dms:setDMSConfigBasicSettings (Sequence Diagram)

This diagram shows the processing that occurs when the user submits the basic settings form for a DMS. The "action" parameter of "setDMSConfigBasicSettings" is mapped to the DMSReqHdlr class via a request action mapping, so its processRequest() is called. It calls setDMSConfigBasicSettings(), which uses the "dmsID" request parameter to look up the WebDMS wrapper object from the cache. The user's login session object is checked to make sure the user has configuration rights. Then a call is made to the DMS CORBA object in the DMS Service to get a snapshot/copy of its current configuration. The request parameters are parsed and used to update the copy of the configuration data, and the DMS object is called again to set the modified configuration data. The PopupSubmissionCloser template is used to close the popup window and display the single command status page, with a "backURL" to allow the user to return to the DMS Details page if they click the Back button from the command status page.

This functionality existed prior to R3B1, but is shown to give context for the parseBasicConfigSettings() diagram, which does contain changes.

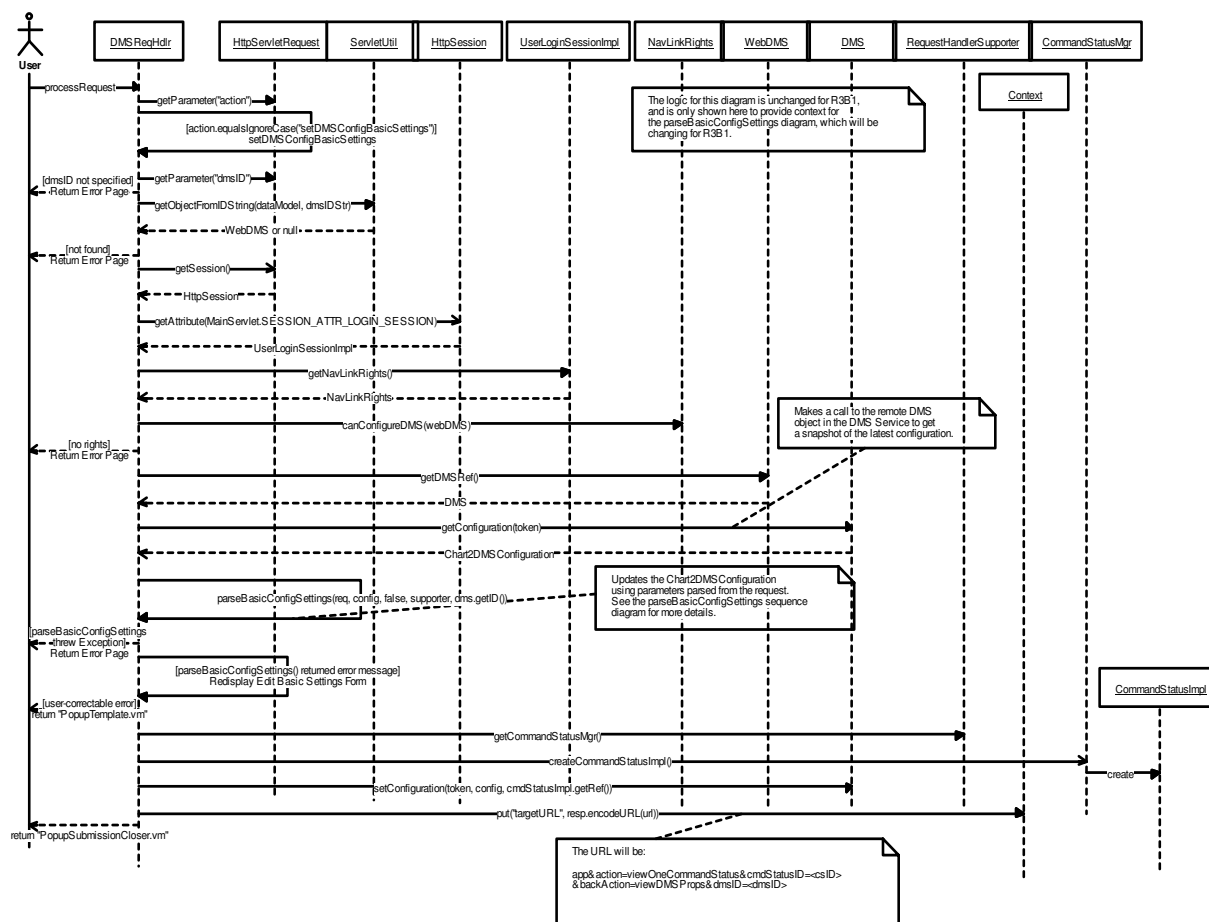


Figure 5-378. chartlite.servlet.dms:setDMSConfigBasicSettings (Sequence Diagram)

5.47.2.12 chartlite.servlet.dms:setDMSTravelRoutes (Sequence Diagram)

This diagram shows the processing to set the travel routes within the DMS configuration. The WebDMS object is retrieved from the cache using the "dmsID" parameter, and the organization-specific user rights are checked. The configuration is queried from the DMS CORBA object. Each route ID specified in the request parameters is added to a list, the route IDs are set back into the configuration, and the DMS is called to set the configuration. The popup submission closer page is returned, telling the browser to close the popup window and redirect the working window to view the View One Command Status page (with the Back button leading to the DMS details page).

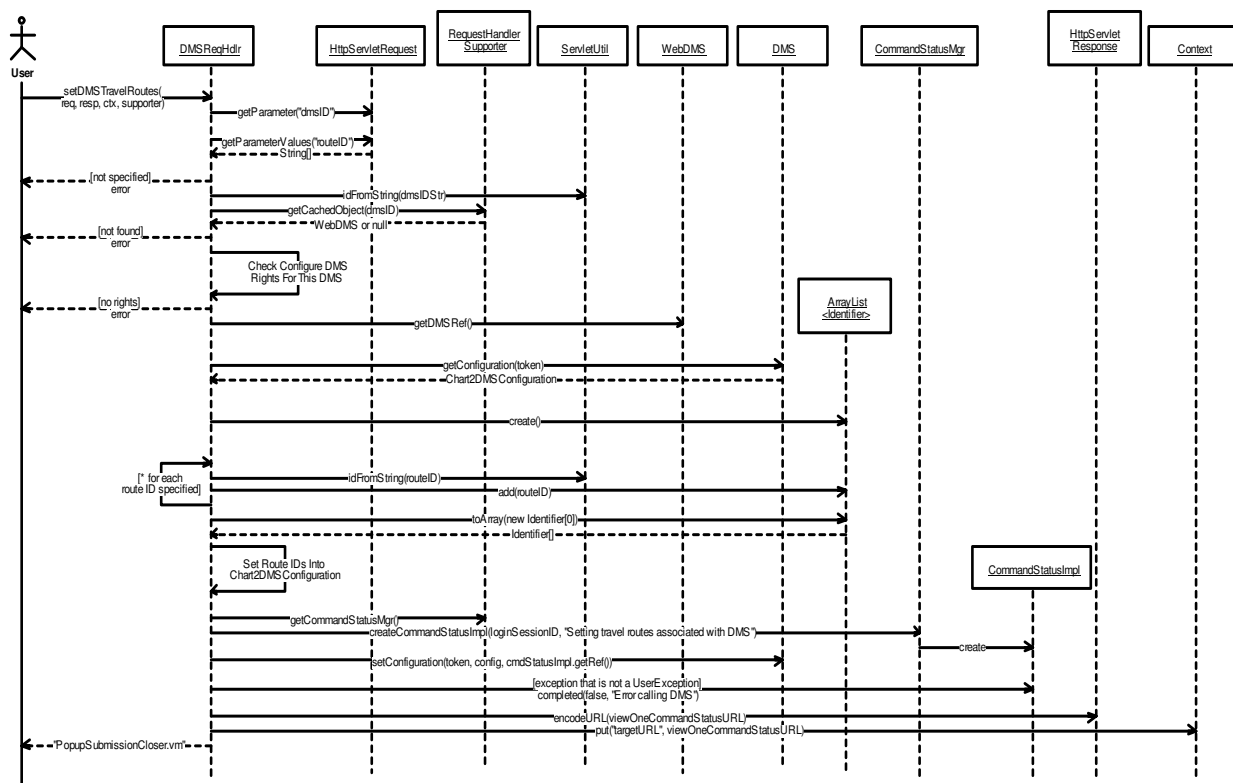


Figure 5-379. chartlite.servlet.dms:setDMSTravelRoutes (Sequence Diagram)

5.47.2.13 chartlite.servlet.dms:setDMSTravelTimeDisplaySchedule (Sequence Diagram)

This diagram shows the processing when the user sets the DMS travel time display schedule. The DMS ID is used to retrieve the WebChart2DMS object from the cache, which is used to check the user's rights for the DMS's organization. The relevant parameters are parsed from the request including the "override" and "useSpecificTimes" flags, and the time ranges if applicable. The Chart2DMS CORBA reference is called to set

the travel time schedule. If successful, the cached Chart2DMSConfiguration is updated to reflect the changed values. Finally the PopupSubmissionCloser template is returned, which will redirect the working window to show the updated DMS Details page before closing the popup window.

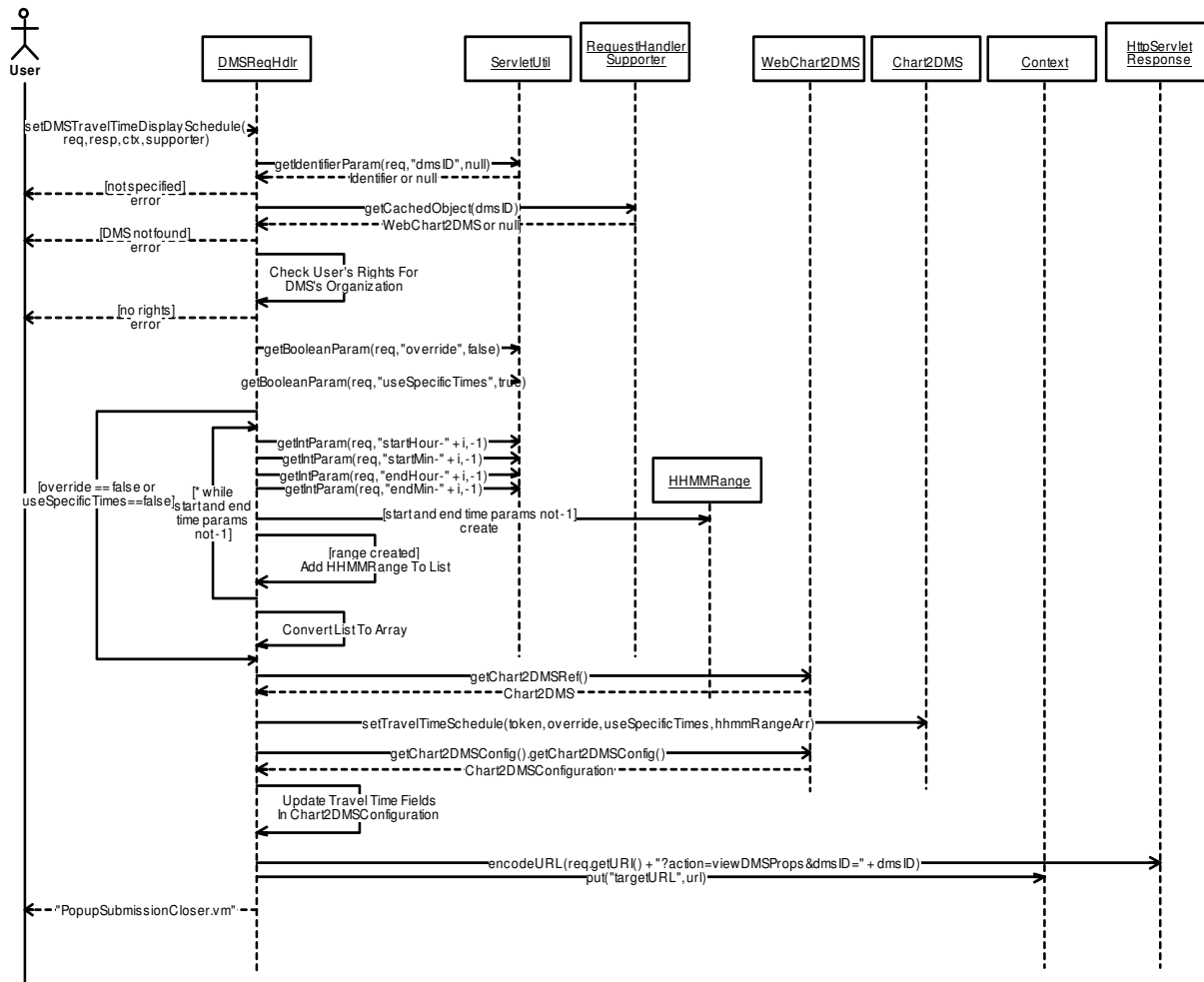


Figure 5-380. chartlite.servlet.dms:setDMSTravelTimeDisplaySchedule (Sequence Diagram)

5.47.2.14 chartlite.servlet.dms:setDMSTravInfoMsgEnabledFlag (Sequence Diagram)

This diagram shows the processing when the user enables or disables a DMS traveler info message. The DMS ID is used to retrieve the WebChart2DMS object from the cache, which is used to check the user's rights for the DMS's organization. The enable/disable flag is parsed from the request and the Chart2DMS CORBA reference is called to set the enable/disable flag. The status is queried from the Chart2DMS to obtain the official enabled message ID, and this status is used to update the cached status in the WebChart2DMS. Finally the response is redirected to show the DMS details page again to show the updated status.

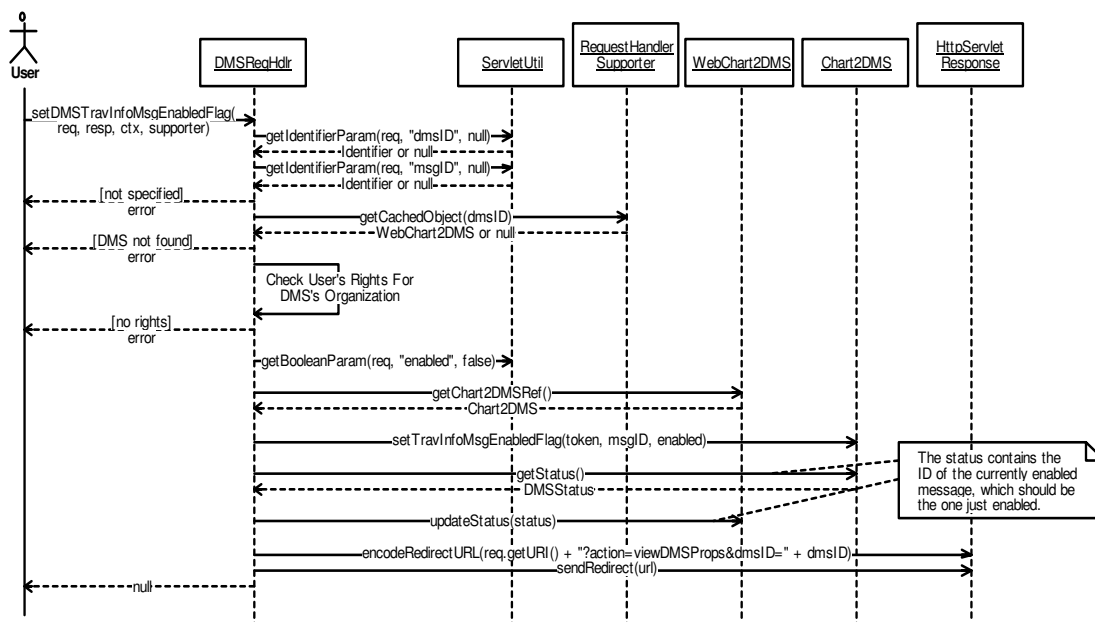


Figure 5-381. chartlite.servlet.dms:setDMSTravInfoMsgEnabledFlag (Sequence Diagram)

5.47.2.15 chartlite.servlet.dms:submitDMSTravInfoMsgForm (Sequence Diagram)

This diagram shows the processing when the form for adding or editing a TravelerInfoMessage is submitted. The parameters for the new TravelerInfoMsg data are parsed from the request, as is the DMS ID and a flag indicating whether it is an edit or add operation. The WebChart2DMS object is obtained from the cache and the user's rights are checked. The DMSTravInfoMsgConfig is queried from the Chart2DMS CORBA object to get a copy of the latest configuration. Depending on whether a message is being edited or added, the message within the copy of the configuration is replaced or the message is appended to the list. The Chart2DMS is called again to set the configuration, and the cached configuration is updated. Finally the Velocity context is populated such that the popup window will cause the DMS Details page to be redisplayed just before the popup closes itself.

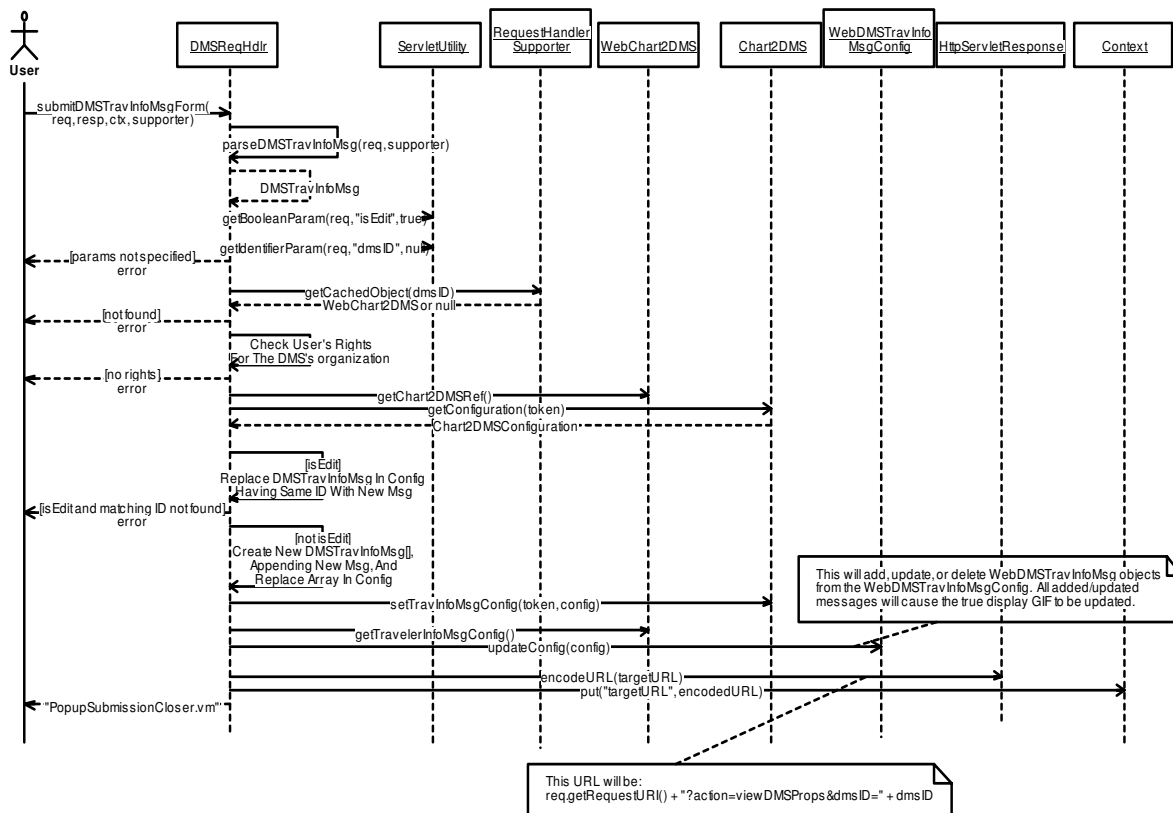


Figure 5-382. chartlite.servlet.dms:submitDMSTravInfoMsgForm (Sequence Diagram)

5.47.2.16 `chartlite.servlet.dms:submitDMSTravInfoMsgTemplateForm` (Sequence Diagram)

This diagram shows the processing that occurs when the DMS template editor form is submitted to create a new template or to commit changes to an existing template. The user's form field values are saved into the editor data. The editor data is then checked for banned words using an existing method (which should not be affected by the presence of the template tags). The message is then checked to see if it fits within the given sign dimensions, by calling `checkDMSEditorDataForFit()`. This method will be changed to call the new method `getFormattedMulti()` which will extract the template data and convert the template data into MULTI (replacing any tags with dummy data). The resulting MULTI is then checked for fit, just as it is already done for the other manual editor types. The message is then checked for spelling (without substituting the template tags). (Most of the tags contain numbers and will be ignored, with the exception being the toll rate time tag, which can be handled by making it an approved word.) If the spell check fails, the spell check page is displayed. Otherwise an attempt is made to create or update the template (see the `createOrUpdateDMSTravInfoMsgTemplate` diagram). If successful, the editor data is removed from the `TempObjectStore` and the `PopupSubmissionCloser` template is displayed, which will refresh the Template List page and close the editor popup. If an error occurred in the message content (i.e., banned words, message didn't fit, or a duplicate template name was used), the editor will be redisplayed showing the error message, which allows the user to fix the problem. If another error occurs, a plain error message page will be displayed.

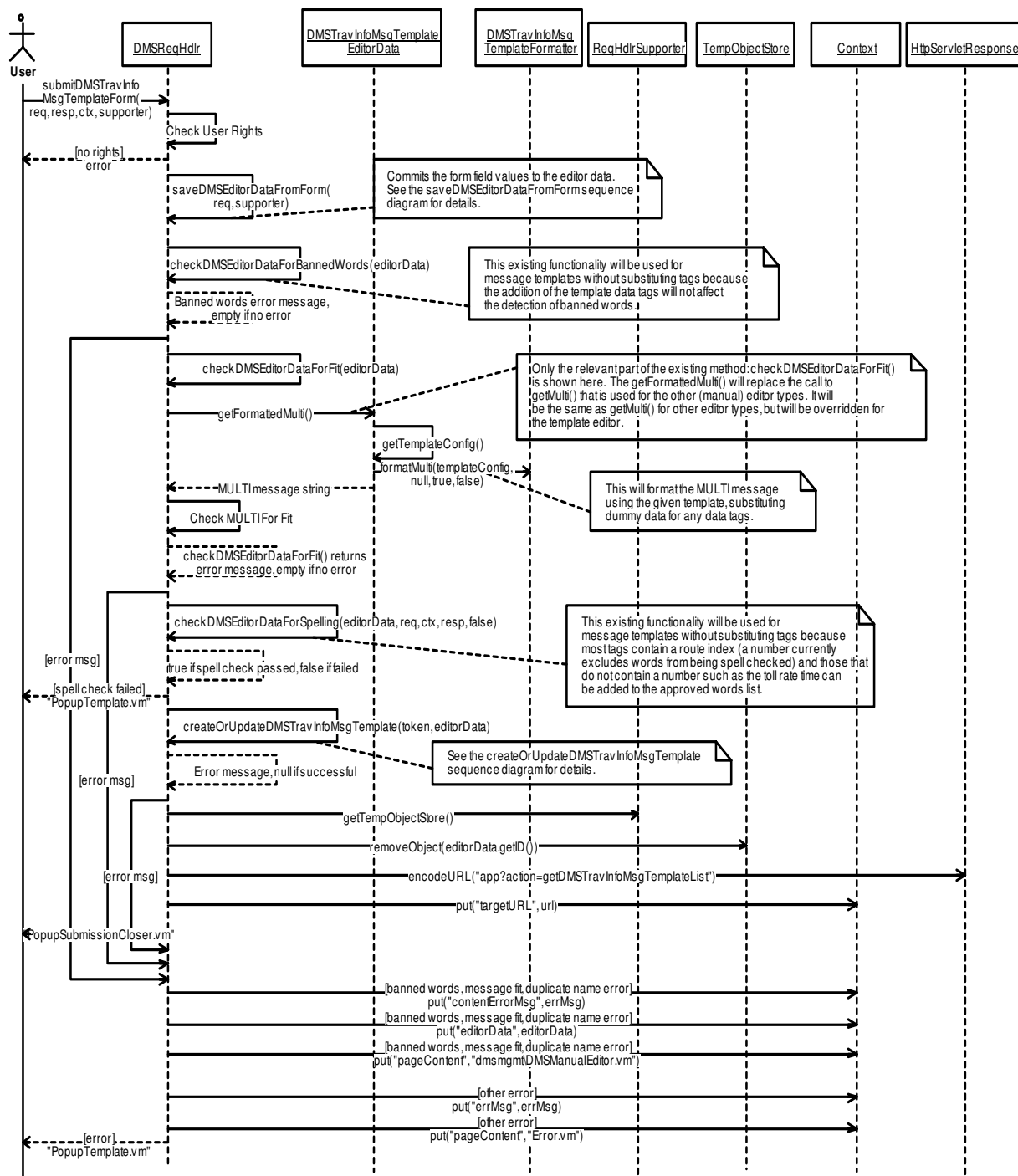


Figure 5-383. chartlite.servlet.dms:submitDMSTravInfoMsgTemplateForm (Sequence Diagram)

5.47.2.17 chartlite.servlet.dms:viewDMSMessageEditorForm (Sequence Diagram)

This diagram shows the processing to display a DMS editor form, including the Traveler

Information Message Template Editor. If the form is being redisplayed and the editor data ID is specified, the DMSEditorData object is retrieved from the temporary object store. If the form is being redisplayed because the message contains a content error such as banned words or a message that doesn't fit on the sign, the content error message is put into the Velocity context to display. If no content error exists, the presence of the "spellCheckResults" parameter indicates that the form is being redisplayed after the resolution of a spell check, and in this case the DMSSpellCheck object will be retrieved from the HTTP session attribute and the message text from the spell check is set into the editor data. If there is no content error or spell check results, the usual processing occurs: if the editor data was not specified then the initial form is being displayed, so the correct type of DMSEditorData is created and added to the temp object store. If the editor data was already found the form is being redisplayed, and in this case any user input parameters specified in the request are set into the editor data before redisplaying the form. Finally the editor data object and the form template name (for either the manual or auto editor) are put into the Velocity context.

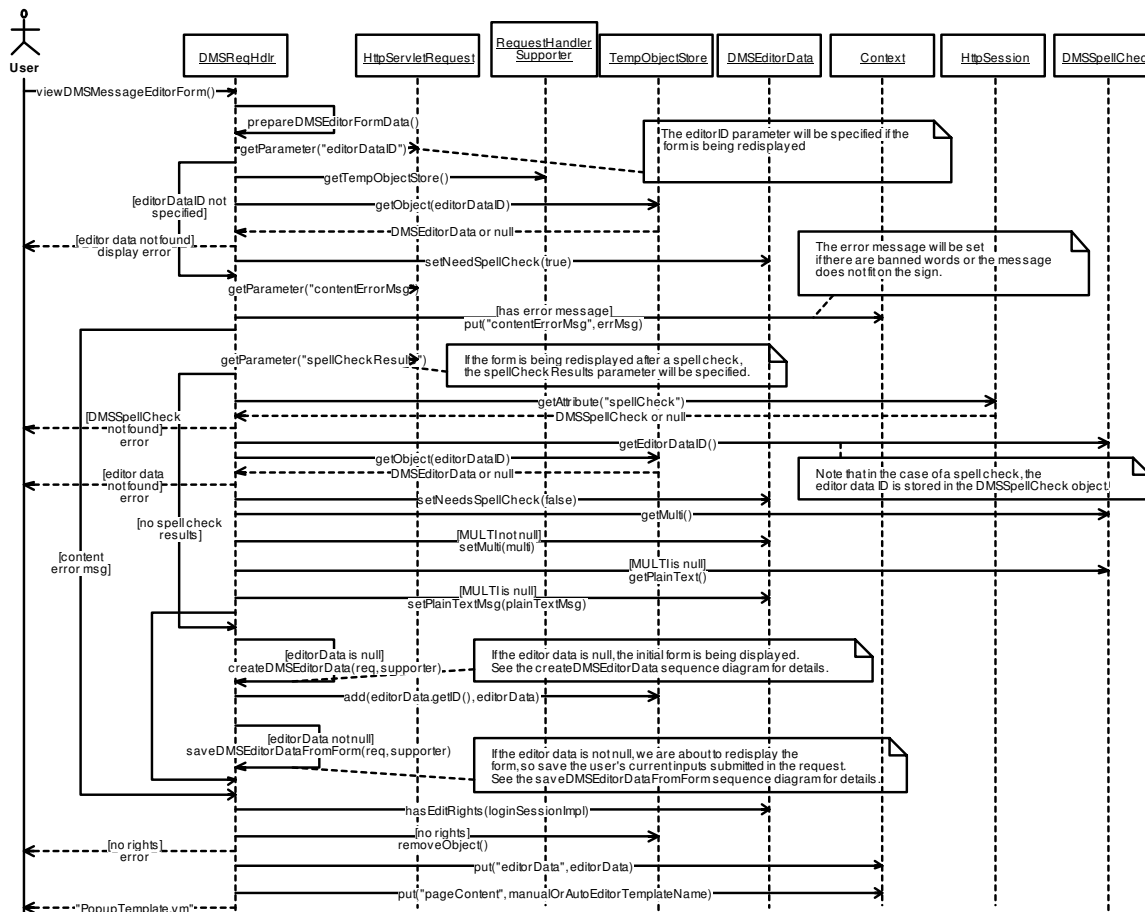


Figure 5-384. chartlite.servlet.dms:viewDMSMessageEditorForm (Sequence Diagram)

5.47.2.18 chartlite.servlet.dms:viewEditDMSTravelRoutesForm (Sequence Diagram)

This diagram shows the processing to display the Edit DMS Travel Routes Form, which allows the user to specify the travel routes available for use by a DMS. The WebDMS object is retrieved from the object cache and the user's rights are checked. If successful the WebTravelRoute objects are retrieved from the cache and sorted, and put into the Velocity context along with the WebDMS. (The DMS's currently selected travel routes can be obtained from the WebDMS's configuration.) The popup template is returned.

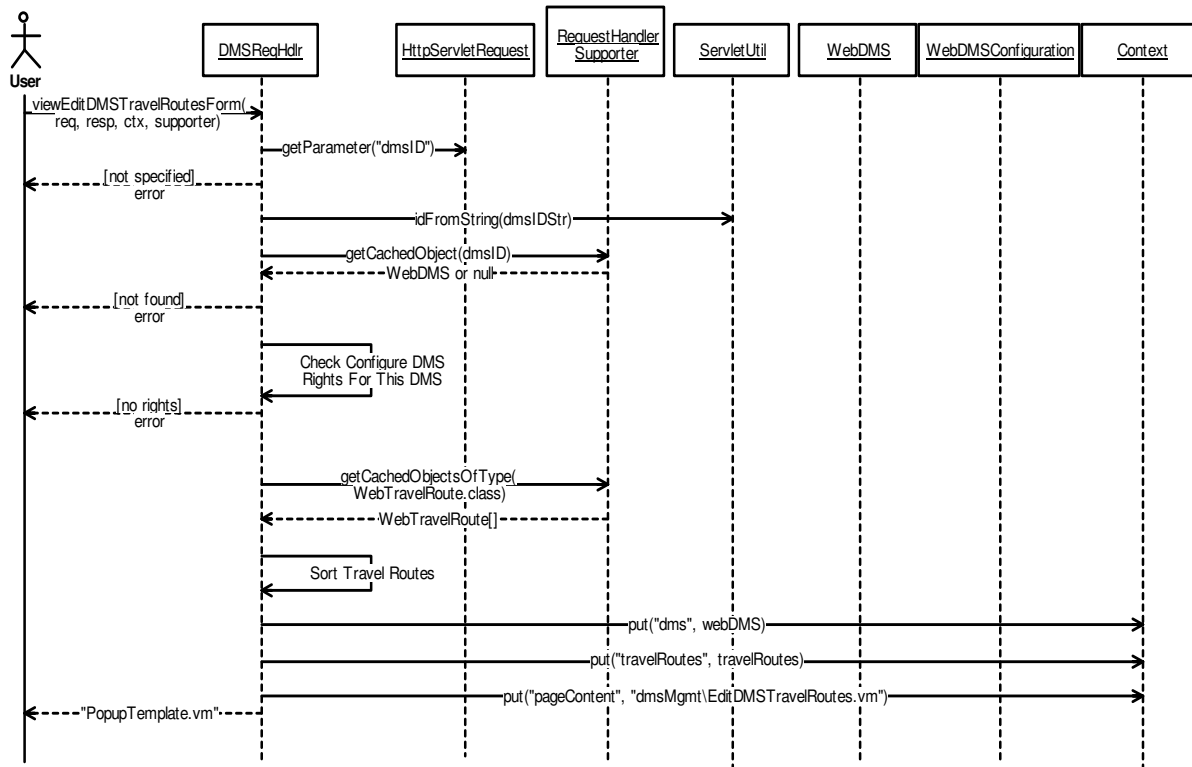


Figure 5-385. chartlite.servlet.dms:viewEditDMSTravelRoutesForm (Sequence Diagram)

5.47.2.19 DMSListSupporter:createDynList (Sequence Diagram)

This diagram shows the processing that occurs when creating a dms dynlist. In R3B3, displayed columns are configurable with defaults and new columns have been added.

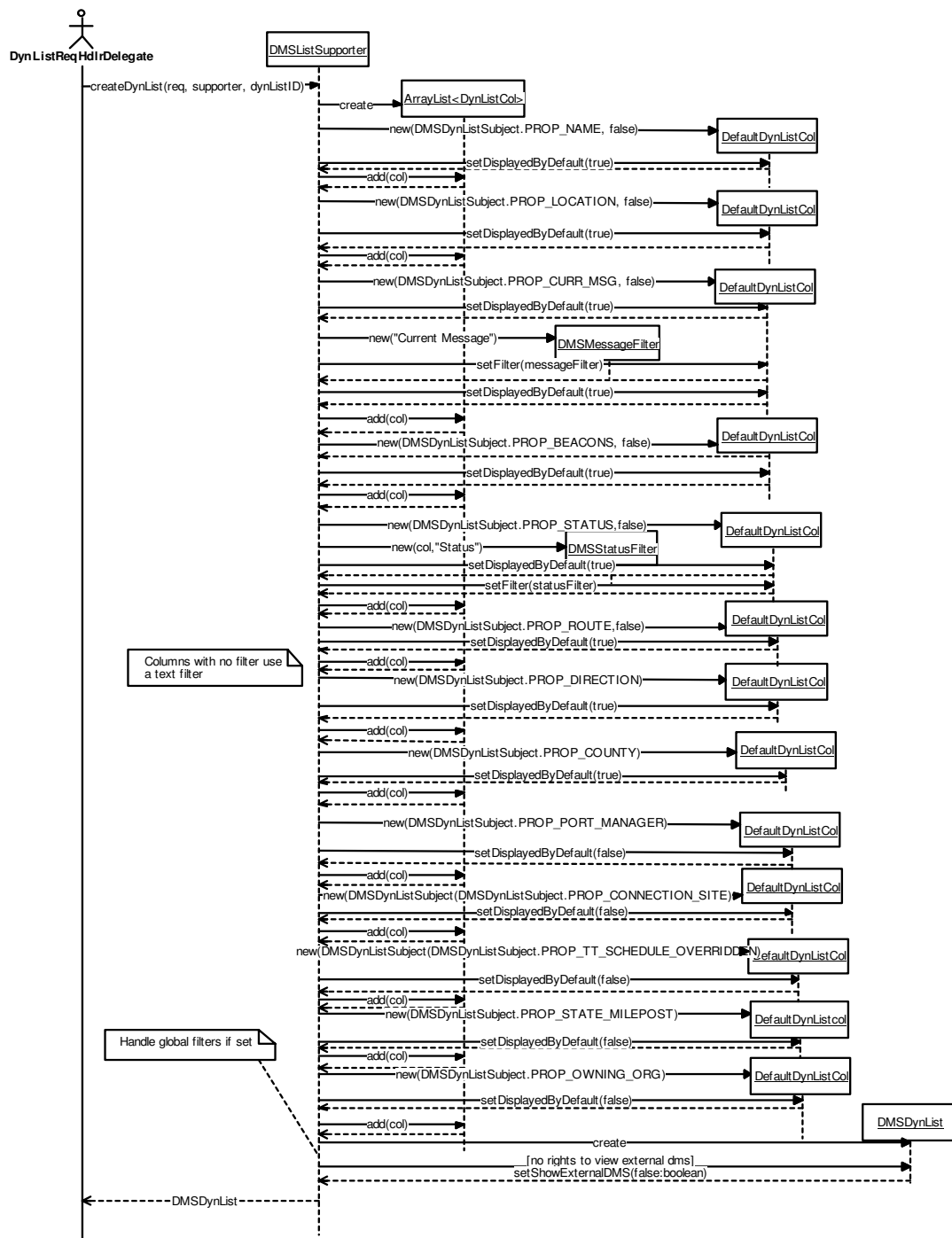


Figure 5-386. DMSListSupporter:createDynList (Sequence Diagram)

5.47.2.20 DMSReqHdlr:getEditDMSLocationForm (Sequence Diagram)

This diagram shows how the Edit DMS Location form is displayed. The formDataID and dmsID parameters are parsed from the request, and one of these should be present (the formDataID if a DMS is being added/copied or the dmsID if editing the location of an existing DMS). If the formDataID is specified, the AddDMSFormData object is retrieved from the TempObjectStore, the form fields are saved into the form data, and a new EditDMSLocationSupporter object is created. If the dms ID is specified, the WebDMS is retrieved from the cache, the user's rights are checked for the given DMS's organization, and the EditDMSLocationSupporter object is created. This object is added to the TempObjectStore and the response is redirected so that the displayEditObjectLocationDataForm request is invoked.

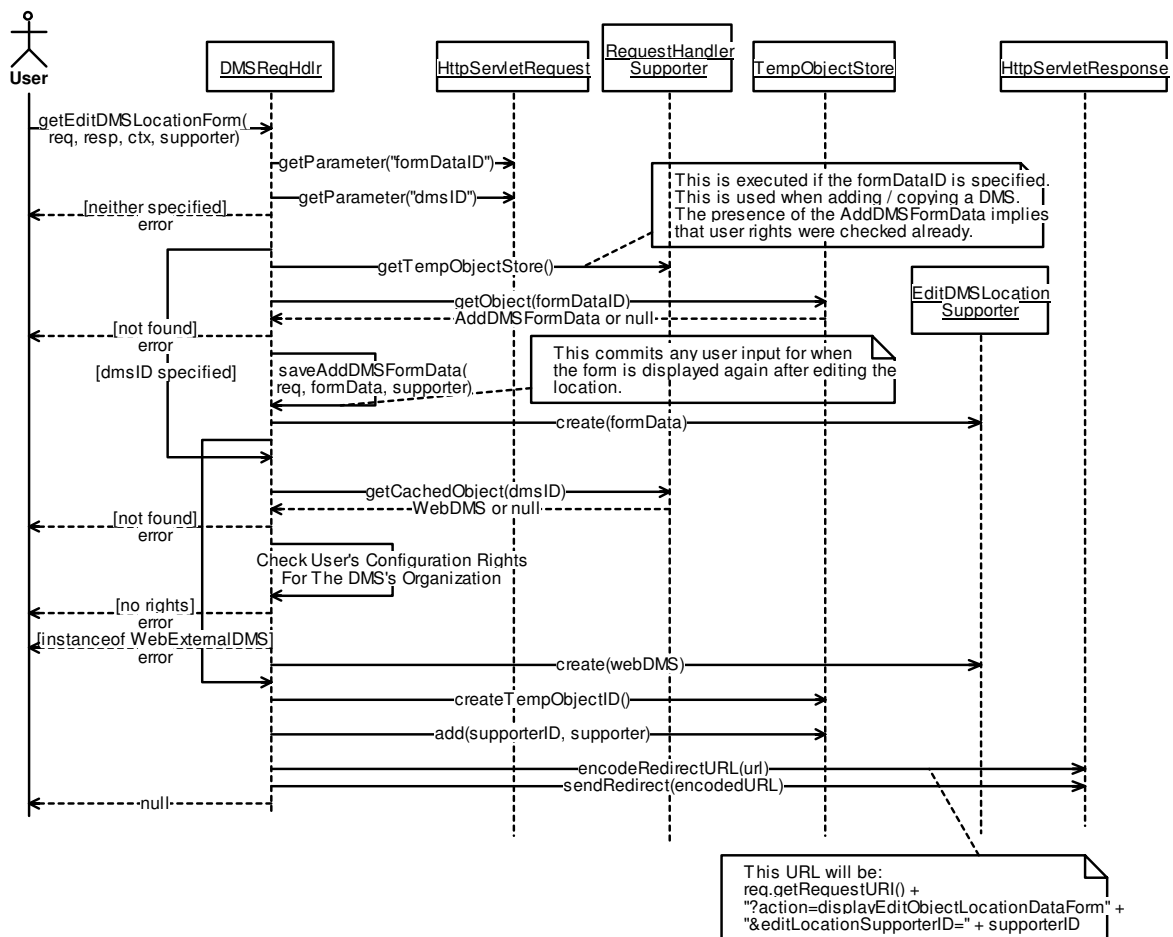


Figure 5-387. DMSReqHdlr:getEditDMSLocationForm (Sequence Diagram)

5.47.2.21 EditDMSLocationSupporter:setObjectLocation (Sequence Diagram)

This diagram shows the processing to save the DMS location when the user submits the Edit Location form. The SpecifyLocationReqHdlr calls the EditDMSLocationSupporter with the location parsed from the request. If the location is being edited while adding / copying a DMS, the configuration is retrieved and altered within the AddDMSFormData object. If it is for an existing DMS, the WebDMS is called to get the DMS reference, the configuration is queried from the DMS, modified within the configuration, and the DMS is called to set the configuration. Since this is an asynchronous command, the URL to view the command status is saved in the EditDMSLocationSupporter and will be passed back in the XML response so that the parent window's URL can be set.

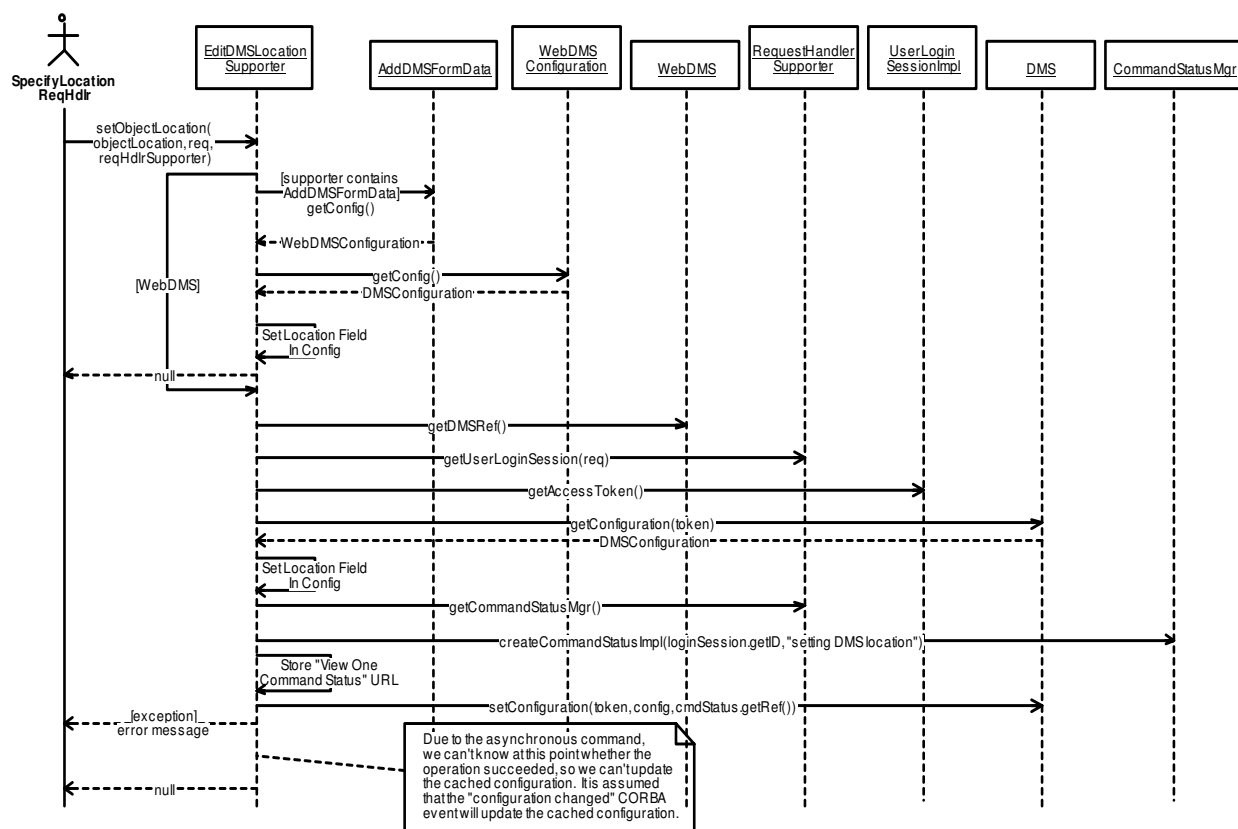


Figure 5-388. EditDMSLocationSupporter:setObjectLocation (Sequence Diagram)

5.48 Chartlite.servlet.alerts

5.48.1 Sequence Diagrams

5.48.1.1 chartlite.servlet.alerts:resolveAlert (Sequence Diagram)

This diagram shows the processing that is done when the servlet receives a request to resolve an alert. This request is issued when the user clicks the resolve button for an alert on the home page, or the user clicks the resolve link on an alert details page. The user's rights are checked, and if they don't have the right required to manage alerts, an error page is returned. The alertID parameter is retrieved from the request, and this is used to find the WebAlert in the servlet's object cache. If the parameter is missing or the alert cannot be found in the object cache, an error page is returned. The getResolutionAction() method is called on the WebAlert. Alert type specific subclasses of WebAlert override this method to provide the query portion of a url used to point the user to right page where they can resolve the alert. After the resolution url is constructed using the query string retrieved from the WebAlert subclass, a redirect to that URL is performed. Following are the resolve actions that each subclass will use:

WebDeviceFailureAlert: viewDMSPProps (DMS device), viewTSSProps (TSS device)

WebDuplicateEventAlert: displayMergeEventSelectTargetForm - pass the events in the alert as the event1 and event2 parameters of the request.

WebManualAlert: viewAlertDetails

WebOpenEventReminderAlert: viewEventDetails - pass the event ID in the alert as the eventID parameter of the request.

WebUnhandledResourcesAlert: getUncontrolledResources

WebExecuteScheduledActionsAlert: The redirect depends on the number of actions in the schedule that fired the alert. If zero actions to execute, viewAlertDetails is the redirect action. If there are more than 1 action, getExecuteScheduledActionsForm is the redirect action. If there is a single Open Event action, viewEventDetails is the redirect action (pass the pending event ID specified in the Open Event action).

WebExternalConnectionAlert: show the connection status page

WebExternalEventAlert: show the details page for the external event WebTollRateAlert: show the details page for the travel route that contains the toll rate source that caused the alert to be generated.

WebTravelTimeAlert: show the details page of the travel route whose travel time exceeded the specified alert travel time.

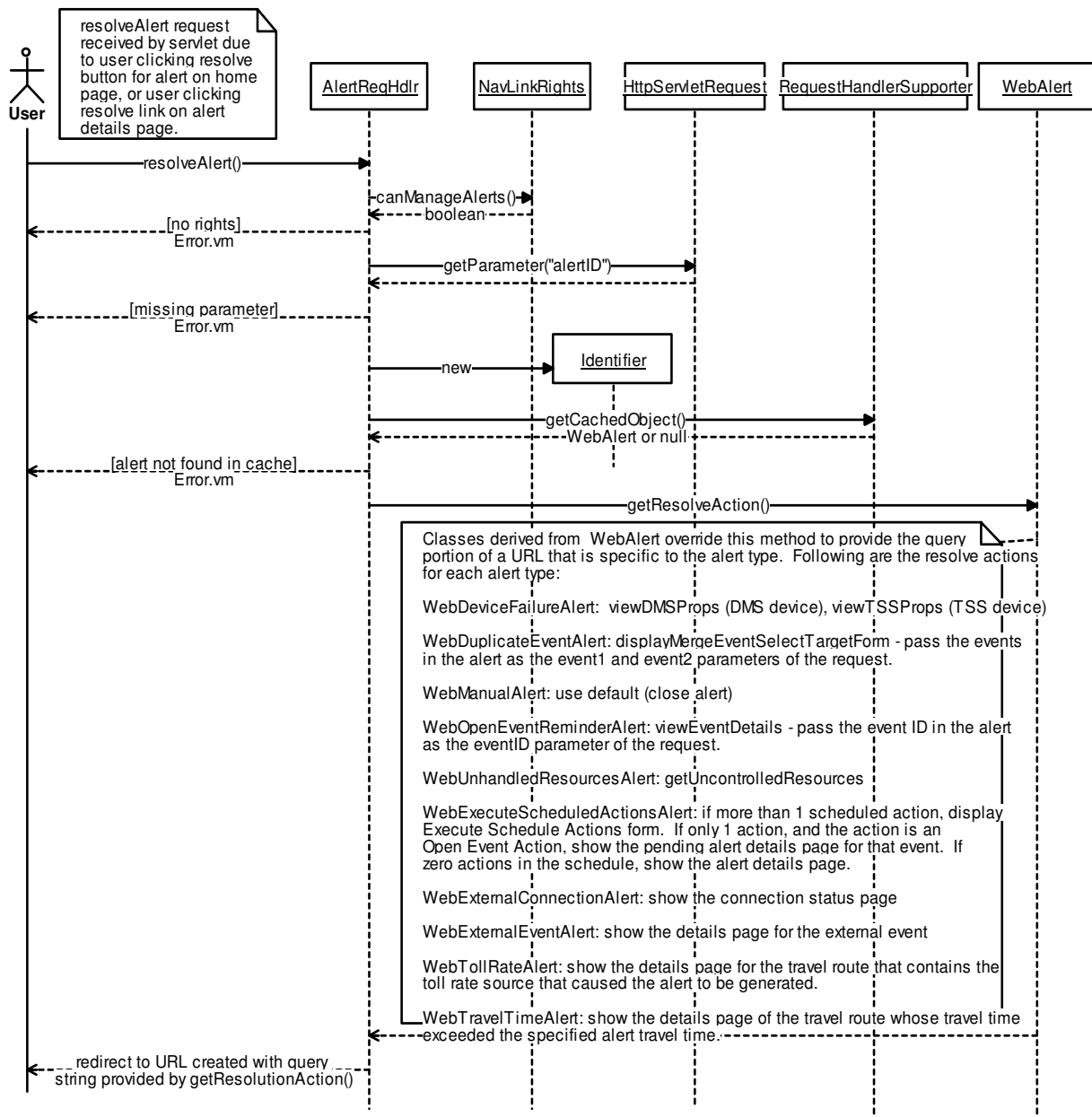


Figure 5-389. chartlite.servlet.alerts:resolveAlert (Sequence Diagram)

5.49 Chartlite.servlet.geoareamgmt

5.49.1 Class Diagrams

5.49.1.1 GUIGeographicAreasServletClasses (Class Diagram)

This class diagram contains classes (servlet) related to geographic areas.

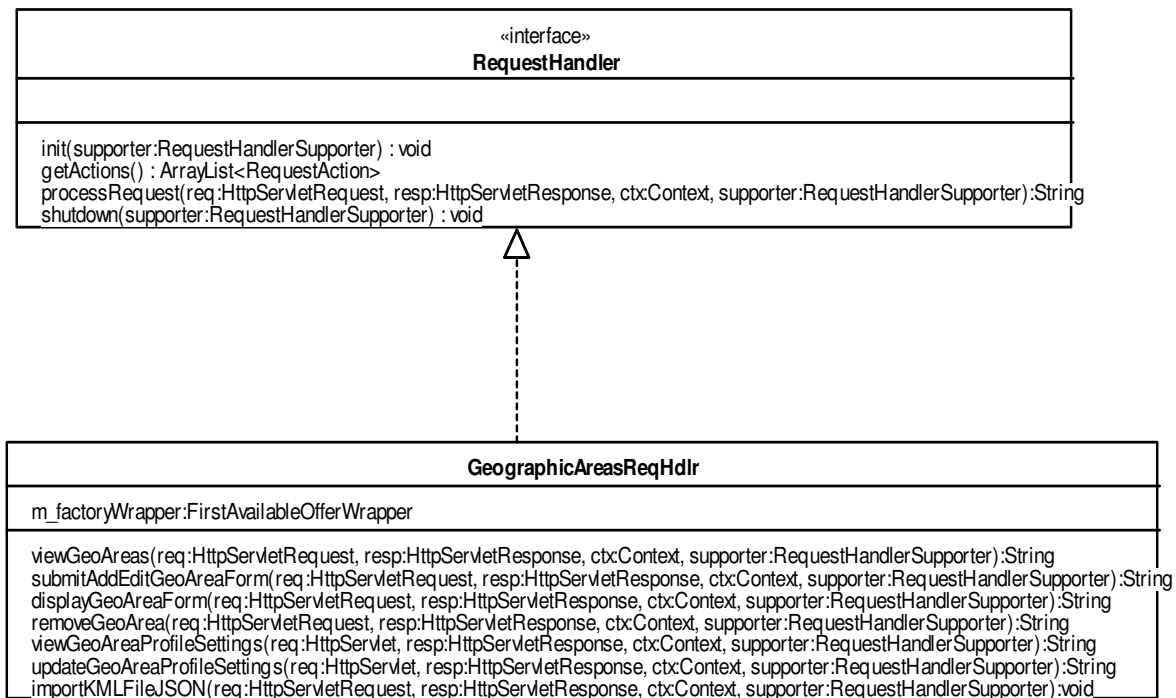


Figure 5-390. GUIGeographicAreasServletClasses (Class Diagram)

5.49.1.1.1 GeographicAreasReqHdlr (Class)

This class handles servlet requests related to geographic areas.

5.49.1.1.2 RequestHandler (Class)

This interface specifies methods that are to be implemented by classes that are used to process requests.

5.49.2 Sequence Diagrams

5.49.2.1 GeographicAreasReqHdlr:displayAddEditGeoAreaForm (Sequence Diagram)

This diagram shows the processing that occurs when an administrator has chosen to add or edit a geographic area. A form is displayed with fields for adding a new area. If the administrator is editing an existing area, it is retrieved from the object cache and placed in the context. If this request is reached via a redirect from the importKML action, points are retrieved from the temp object store and placed in the context.

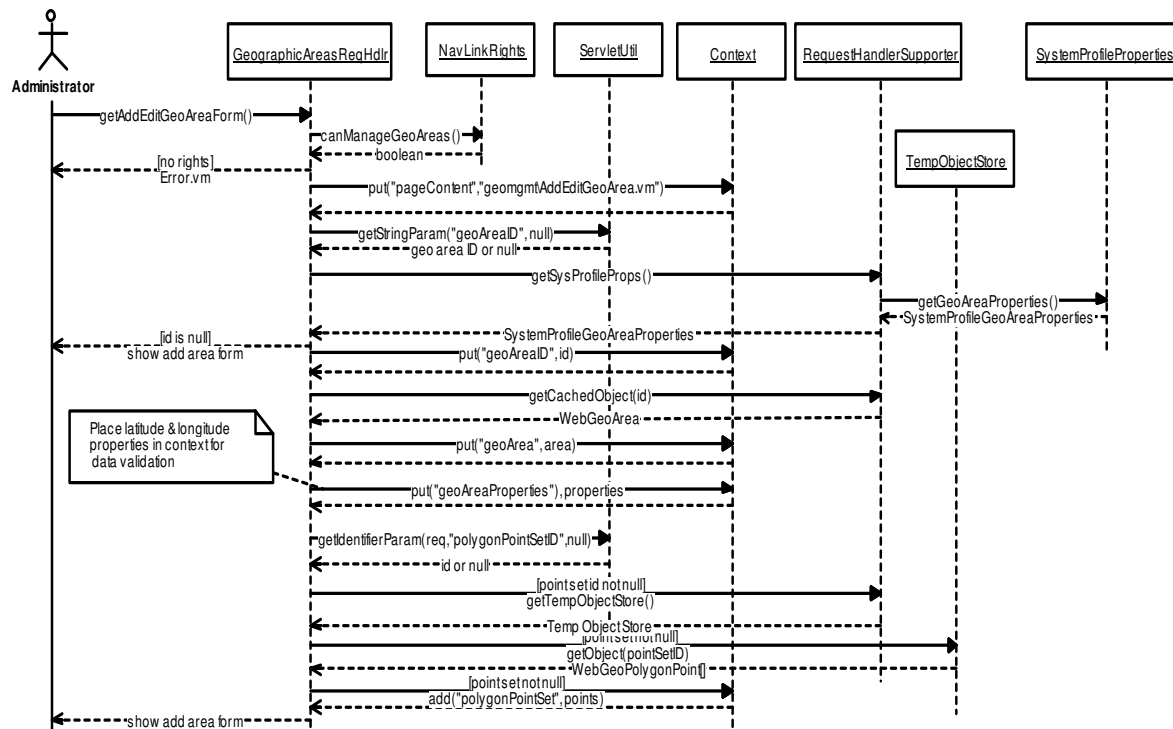


Figure 5-391. GeographicAreasReqHdlr:displayAddEditGeoAreaForm (Sequence Diagram)

5.49.2.2 GeographicAreasReqHdlr:importKMLFileJSON (Sequence Diagram)

This diagram shows the processing that occurs when an administrator has chosen to import points from a KML file. The file is submitted via an ajax request from the add area form. The xml file is read and all coordinate pairs are placed in JSON objects. An array of JSON objects is sent to the user for population of the coordinate pairs on the form using javascript.

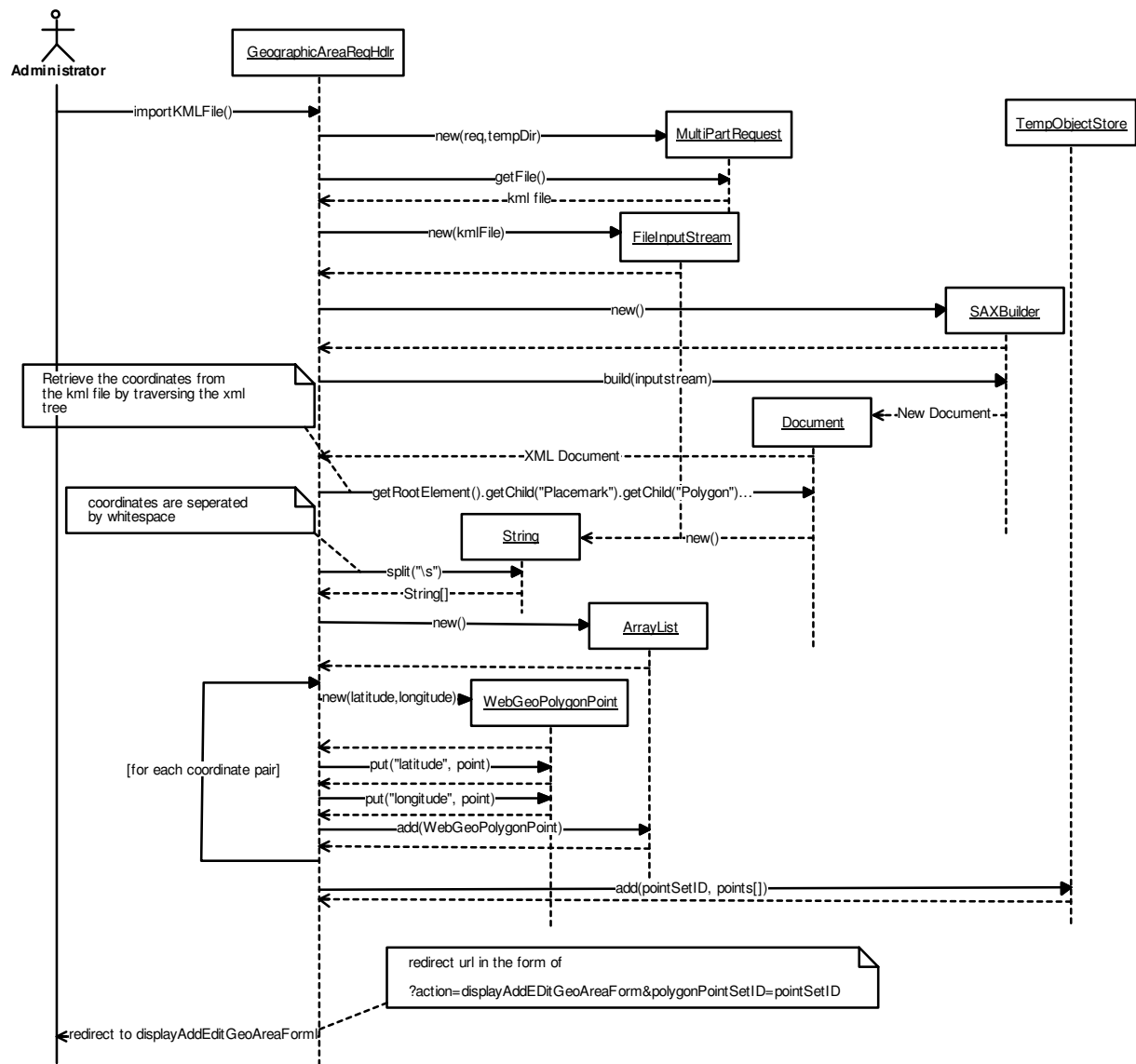


Figure 5-392. GeographicAreasReqHdlr:importKMLFileJSON (Sequence Diagram)

5.49.2.3 GeographicAreasReqHdlr:removeGeoArea (Sequence Diagram)

This diagram shows the processing that occurs when an administrator removes a geographic area. Geographic areas cannot be removed unless unreferenced in the system.

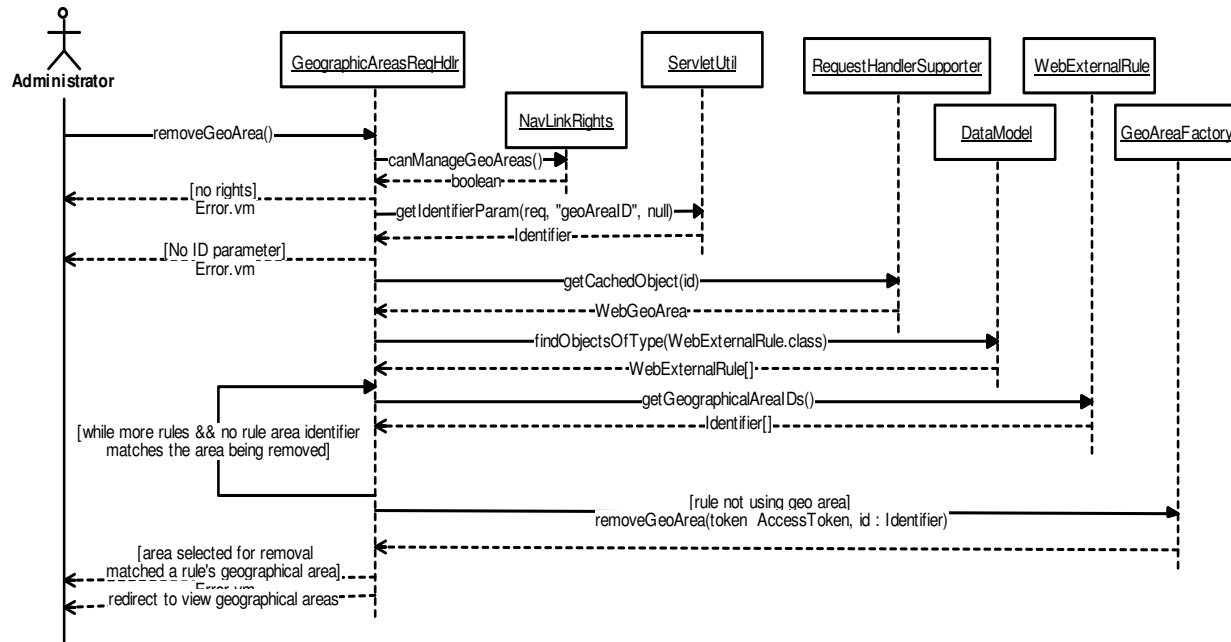


Figure 5-393. GeographicAreasReqHdlr:removeGeoArea (Sequence Diagram)

5.49.2.4 GeographicAreasReqHdlr:submitAddEditGeoAreaForm (Sequence Diagram)

This diagram show the processing that occurs when an administrator has submitted a request to add/edit a geographic area.

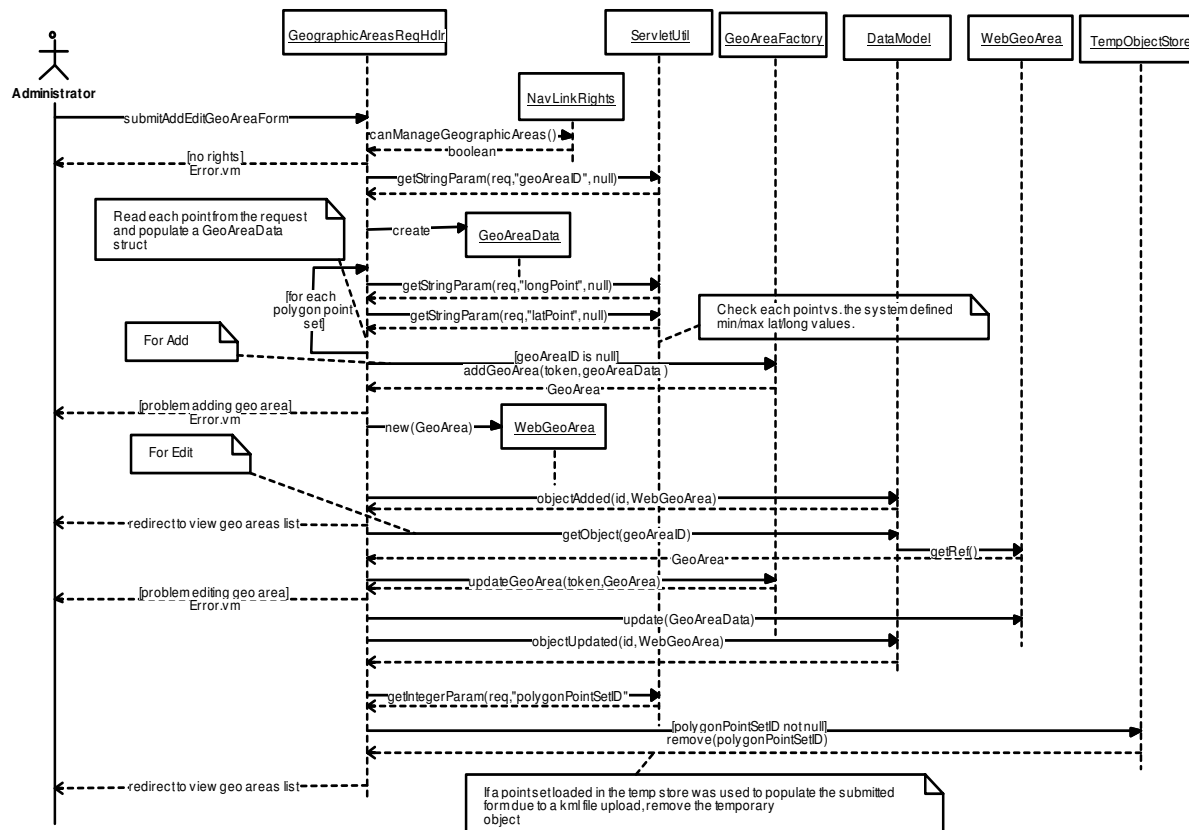


Figure 5-394. GeographicAreasReqHdlr:submitAddEditGeoAreaForm (Sequence Diagram)

5.50 Chartlite.servlet.travelroutes

5.50.1 Class Diagrams

5.50.1.1 GUITravelRouteServletClasses (Class Diagram)

This diagram shows classes used by the servlet to handle requests related to travel routes.

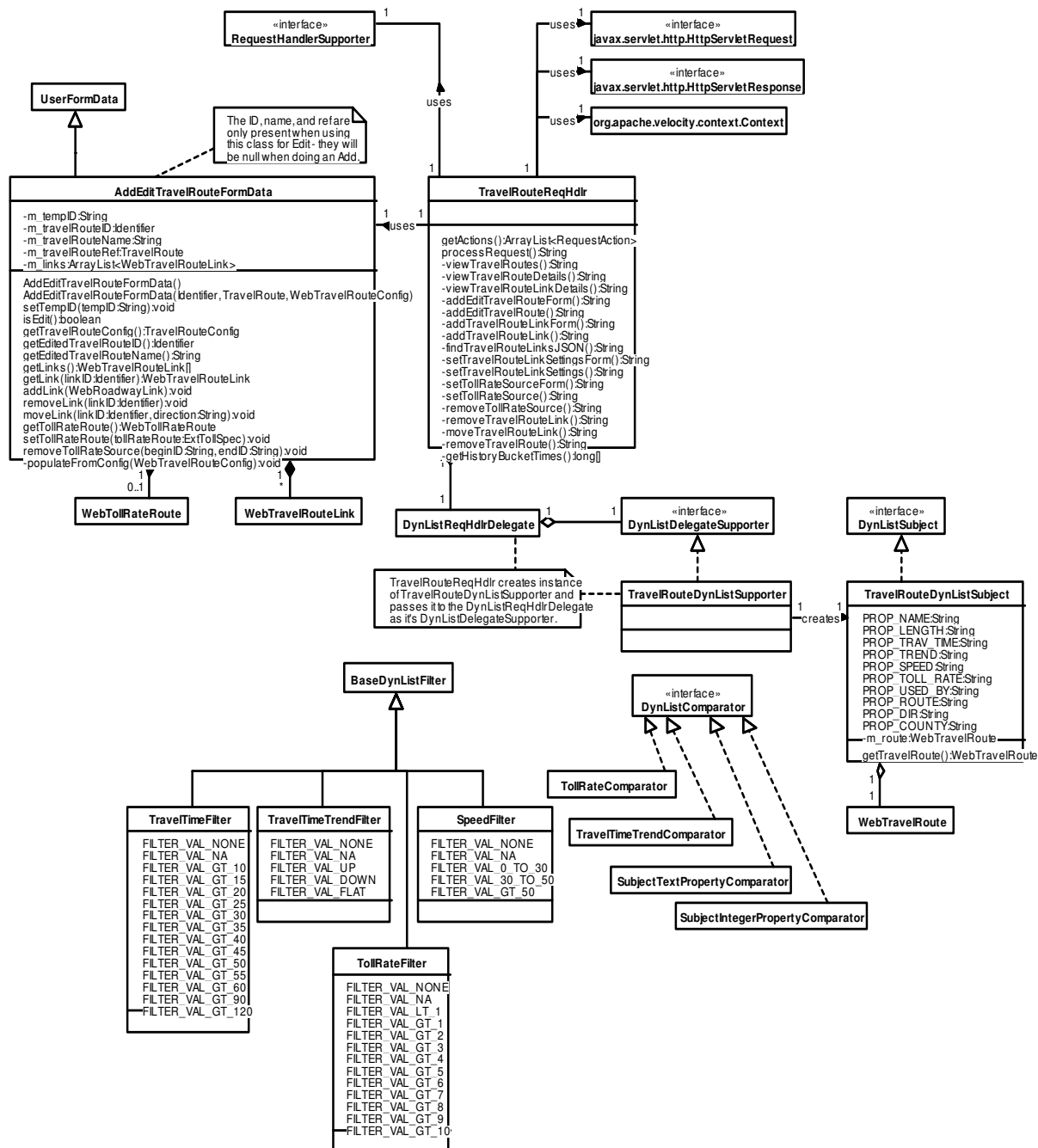


Figure 5-395. GUITravelRouteServletClasses (Class Diagram)

5.50.1.1.1 AddEditTravelRouteFormData (Class)

This class holds data that appears on the add/edit travel route form so the data can be temporarily persisted while the user accesses auxillary forms that may be used.

5.50.1.1.2 BaseDynListFilter (Class)

This abstract class provides a base implementation of the DynListFilter interface.

5.50.1.1.3 DynListComparator (Class)

This interface is implemented by classes that are used to sort dynamic lists.

5.50.1.1.4 DynListDelegateSupporter (Class)

This interface contains functionality to support the DynListReqHdlrDelegate

5.50.1.1.5 DynListReqHdlrDelegate (Class)

This class helps request handlers support dynamic lists. Requests to view, sort, or filter dynamic lists can be passed from a request handler to this class, provided the URL used for the requests contain parameters required by this class, such as the id of the list, the property name, and/or the filter value.

5.50.1.1.6 DynListSubject (Class)

This interface is implemented by classes that wish to be capable of being displayed in a dynamic list.

5.50.1.1.7 javax.servlet.http.HttpServletRequest (Class)

Provides information about a request made to an HTTP servlet.

5.50.1.1.8 javax.servlet.http.HttpServletResponse (Class)

Provides a way for an HTTP servlet to send a response.

5.50.1.1.9 org.apache.velocity.context.Context (Class)

This class is used to allow an application to provide "named" data to a template so the data can be used when rendering the template.

5.50.1.1.10 RequestHandlerSupporter (Class)

This interface is implemented by any class that can provide access to objects or methods that are helpful to request handlers.

5.50.1.1.11 SpeedFilter (Class)

This class is a filter for columns that contain Speed within a dynamic list.

5.50.1.1.12 SubjectIntegerPropertyComparator (Class)

This class is a dyn list comparator that can be used to sort columns that contain integer values.

5.50.1.1.13SubjectTextPropertyComparator (Class)

This class provides an implementation of the DynListComparator interface which compares subjects based on the values they supply for the property supplied to this class during construction. A case insensitive text comparison is done on the values, and multiple value columns are supported.

5.50.1.1.14TollRateComparator (Class)

This class is a comparator used to compare toll rate values that appear in a dyn list.

5.50.1.1.15TollRateFilter (Class)

This class is a filter used to filter items in a dyn list based on toll rate values.

5.50.1.1.16TravelRouteDynListSubject (Class)

This class is a wrapper that allows a WebTravelRoute to be displayed in a dynamic list.

5.50.1.1.17TravelRouteDynListSupporter (Class)

This class provides methods used by the DynListReqHdlrDelegate to customize the dynamic list to display travel routes.

5.50.1.1.18TravelRouteReqHdlr (Class)

This class processes requests related to travel routes.

5.50.1.1.19TravelTimeFilter (Class)

This class is used to filter a dynamic list based on travel time values.

5.50.1.1.20TravelTimeTrendComparator (Class)

This class is a comparator used to sort a dynamic list based on travel time trend.

5.50.1.1.21TravelTimeTrendFilter (Class)

This class is used to filter a dynamic list based on travel time trend values.

5.50.1.1.22UserFormData (Class)

This class is used to store form data between requests while a user is editing a complex form, and provides convenience methods for parsing the values from the request.

5.50.1.1.23WebTollRateRoute (Class)

This class is a wrapper for data from a TollRateRouteInfo object obtained from the travel route factory. It is used to cache data pertaining to a toll rate route in the GUI's data model. (A Toll Rate Route can be used as the toll rate source for a travel route).

5.50.1.1.24WebTravelRoute (Class)

This class is a wrapper for a CORBA TravelRoute object and is used to cache data pertaining to a travel route in the GUI's data model.

5.50.1.1.25WebTravelRouteLink (Class)

This class holds data pertaining to a link that is included in a travel route.

5.50.2 Sequence Diagrams

5.50.2.1 TravelRouteDynListSupporter:createDynList (Sequence Diagram)

This diagram shows the processing used to create a travel route dynamic list. Each column of the list is created, along with comparators and filters used for each column. Those shown that do not create a filter are not filterable. Those columns that do not show construction of a comparator use the default text comparator that will be created by the DefaultDynListCol class. After creating the columns, they are passed to the constructor of a DefaultDynList, and the DefaultDynList is returned.

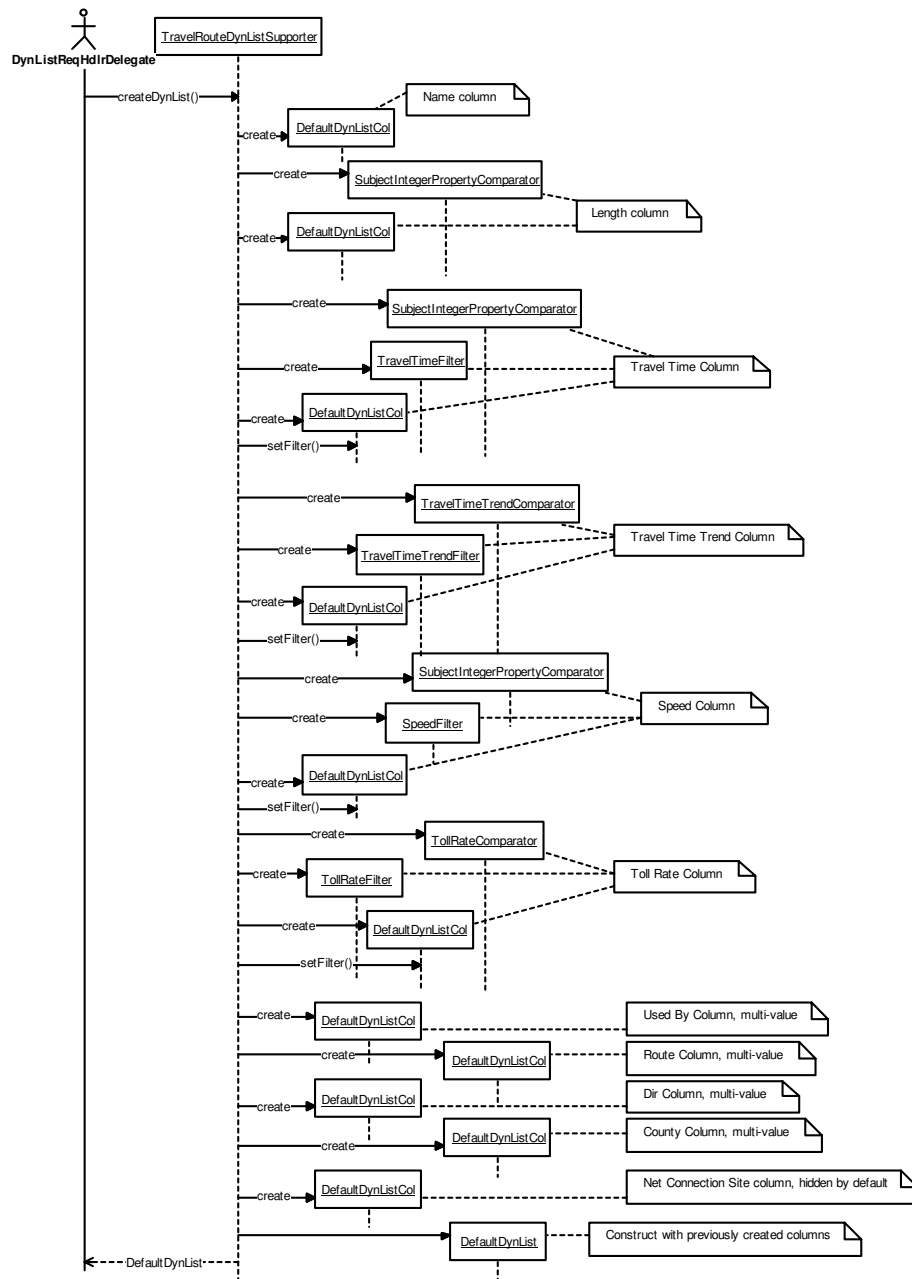


Figure 5-396. TravelRouteDynListSupporter:createDynList (Sequence Diagram)

5.50.2.2 TravelRouteDynListSupporter:getDynListSubjects (Sequence Diagram)

This diagram shows the processing performed when the TravelRouteDynListSupporter is called to get the dyn list subjects. It retrieves all WebTravelRoute objects from the object cache and wraps each with a TravelRouteDynListSubject, then returns this list of subjects.

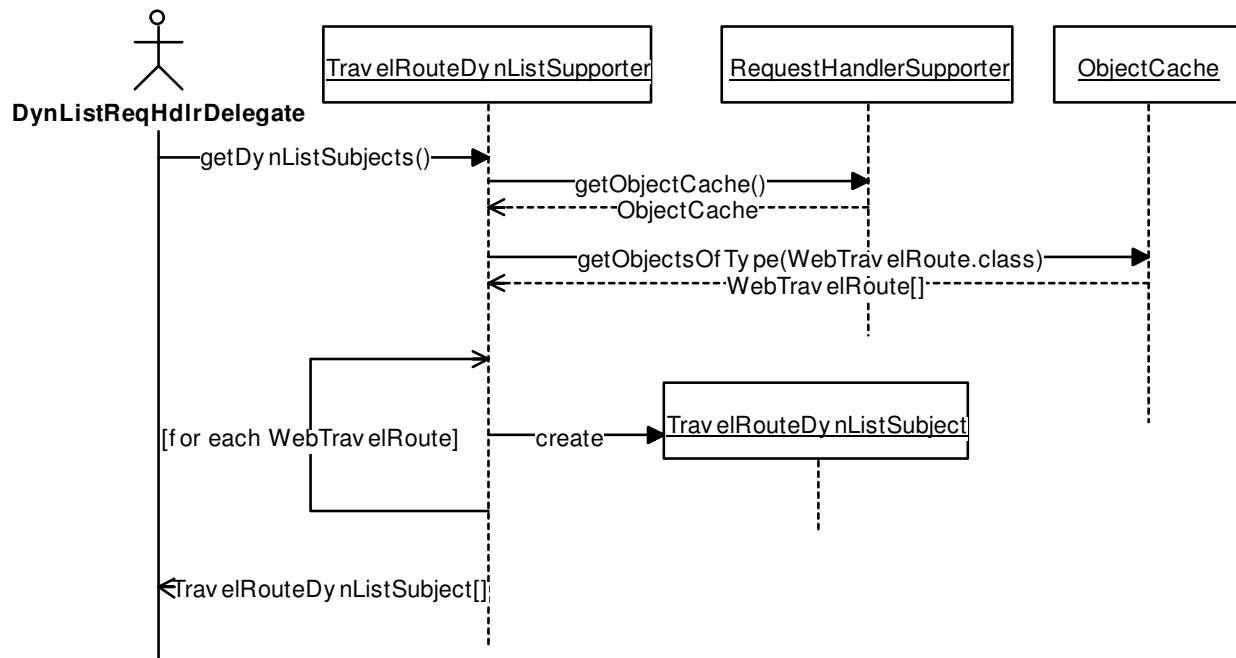


Figure 5-397. TravelRouteDynListSupporter:getDynListSubjects (Sequence Diagram)

5.50.2.3 TravelRouteReqHdlr:addEditTravelRoute (Sequence Diagram)

This diagram shows the processing that takes place when the AddEditTravelRoute form is submitted. The user's rights are checked to make sure they are permitted to manage travel routes. If not, an error is placed in the form data object and the form is redisplayed with the error message. The form data object is then populated with the data included in the request parameters, and the form data object is called to construct a TravelRouteConfig object from the data. If any required data is missing or data is invalid, the config object will not be created and an error message will be set in the form data object. If the operation is an "Add", the travel route factory is retrieved from the GUI cache. If any error has occurred at this point, the add/edit form will be redisplayed and show an error message.

If performing an edit, the travel route factory being edited is retrieved from the GUI cache, its corba object reference is obtained, and the corba object is called to set the configuration. If performing an edit, the factory's corba object reference is obtained, the factory is called to add the travel route, and the travel route is stored in the GUI cache.

After the processing is complete, the user is shown the travel route list where the new travel route will appear (or the edited values will appear if editing an existing travel route).

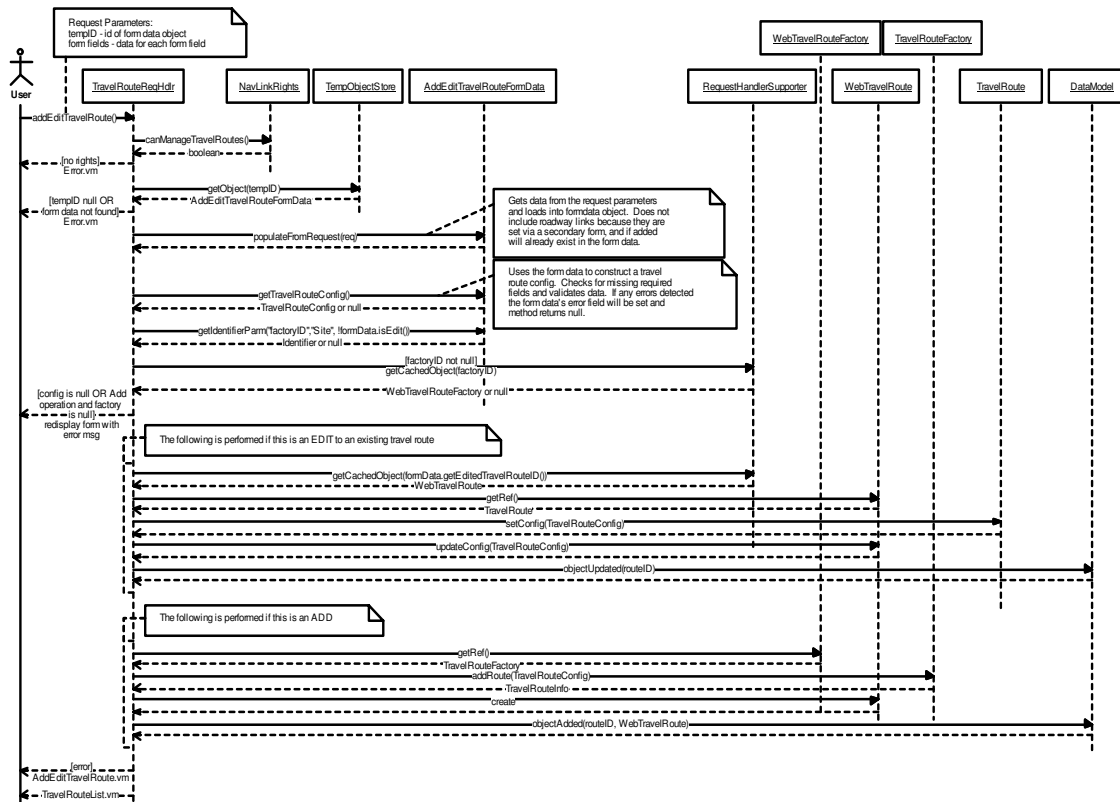


Figure 5-398. TravelRouteReqHdlr:addEditTravelRoute (Sequence Diagram)

5.50.2.4 TravelRouteReqHdlr:addEditTravelRouteForm (Sequence Diagram)

This diagram shows the processing that is performed if a user requests to add or edit a travel route. Their user rights are checked to make sure they are permitted to manage travel routes. If not an error is returned. If a tempID parameter is present in the request, this indicates the add/edit is already in progress and the add/edit form is being redisplayed. This will be the case if the user needs to navigate to a secondary form as part of the add/edit. The AddEditTravelRouteFormData object will be retrieved from the TempObjectStore in this case. If it's not found, an error will be shown.

If a tempID is not present in the request, the routeID will be retrieved from the request. If the route ID is present, this indicates an edit operation is requested. Otherwise an add operation is being requested. When an edit operation is requested, the WebTravelRoute being edited is retrieved from the object cache and its data is used to construct a new AddEditTravelRouteFormData object (which is eventually used to populate the form). If an add operation is being performed, the default AddEditTravelRouteFormData constructor is used (which will result in an initially empty form). In either case, the AddEditTravelRouteData object is stored in the TempObjectStore so it can be accessed during interim submits of the form data as well as when the final submit is done.

Various objects are obtained from the object cache that are used on the form to provide user selections. This includes travel route factories, operations centers, notification groups, states, and counties. Each list of objects is placed in the velocity context so they can be accessed from within the form. The AddEditTravelRoute.vm form is returned and displayed to the user, infused with data it obtains from the velocity context.

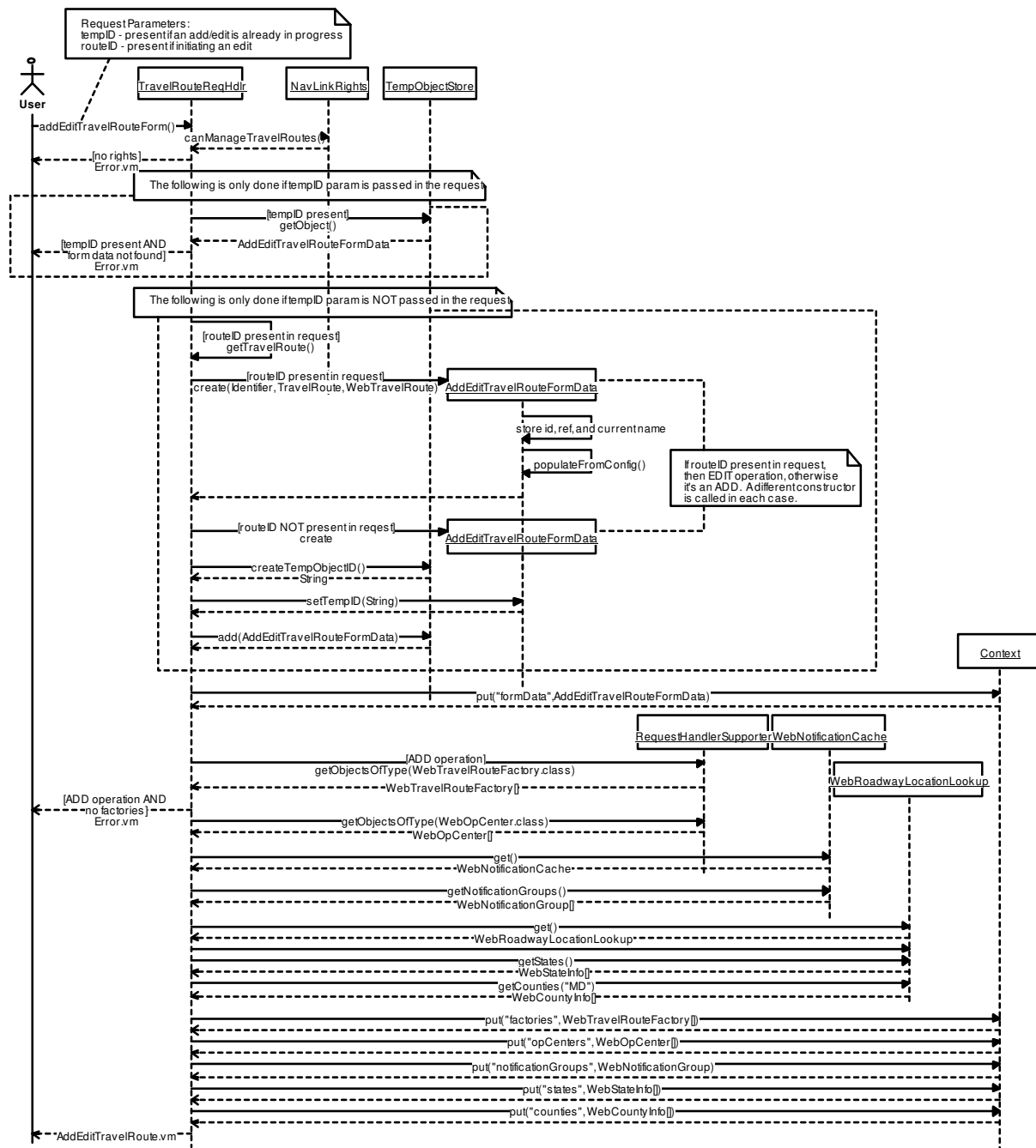


Figure 5-399. TravelRouteReqHdlr:addEditTravelRouteForm (Sequence Diagram)

5.50.2.5 TravelRouteReqHdlr:addTravelRouteLink (Sequence Diagram)

This diagram shows the processing that is performed when the user submits the add link form. This form can be used as part of adding/editing a travel route, or from the details page as a single operation to add links to a travel route. When used as part of the add/edit travel route operation, a tempID will be present in the request parameters. It will be used to find the form data object in the temp object store. A WebRoadwayLink object for each linkID contained in the request parameters will be obtained from the RoadwayLinkManager and will be added to the form data. The user will be redirected to the add/edit travel route form where the newly added links will appear.

If the form was invoked from the travel route details page, the WebTravelRoute object is obtained from the GUI cache and the Travel Route object ref is used to obtain the current configuration data from the server. A RouteLink object is created for each of the links being added, and the TravelRouteConfig object is updated. The TravelRoute's setTravelTimeConfig method is called, and the WebTravelRoute is called to update the configuration data in the cached object. The user is then redirected to the travel route details page.

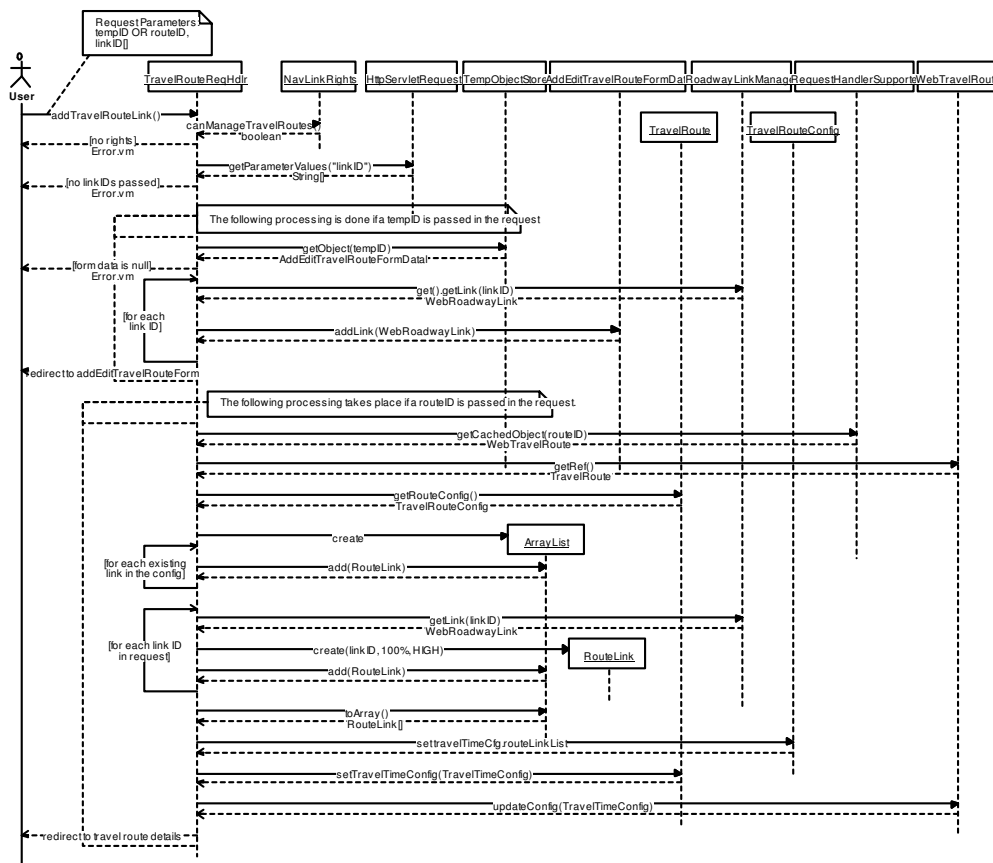


Figure 5-400. TravelRouteReqHdlr:addTravelRouteLink (Sequence Diagram)

5.50.2.6 TravelRouteReqHdlr:addTravelRouteLinkForm (Sequence Diagram)

This diagram shows the processing that is performed when the user chooses to select links to add to a travel route. The form can be accessed while adding/editing a travel route, or from the travel route details page. When accessed as part of an add/edit operation, the tempID parameter will be present in the request and will be used to retrieve the associated form data object and store the request parameters in it (so they can be redisplayed on the add/edit form after the user finishes selecting links). Also, the list of links currently contained in the travel route are obtained from the form data.

If the user accesses this form from the travel route details page, a routeID will be present in the request parameters and the WebTravelRoute will be obtained from the object cache. The links contained in the travel route will be obtained from the WebTravelRoute object.

If there are 1 or more links already added to the travel route (or the travel route being added/edited), the roadway link manager is called to obtain suggested next links and they are placed in the context, along with the last existing link (which will be displayed as the prior link). The lists of states, counties, route types, and route numbers that exist in one or more links are obtained from the roadway link manager and placed in the context to be used to populate select lists. The SelectLink.vm template is returned and is used to dynamically generate a web page using data in the context.

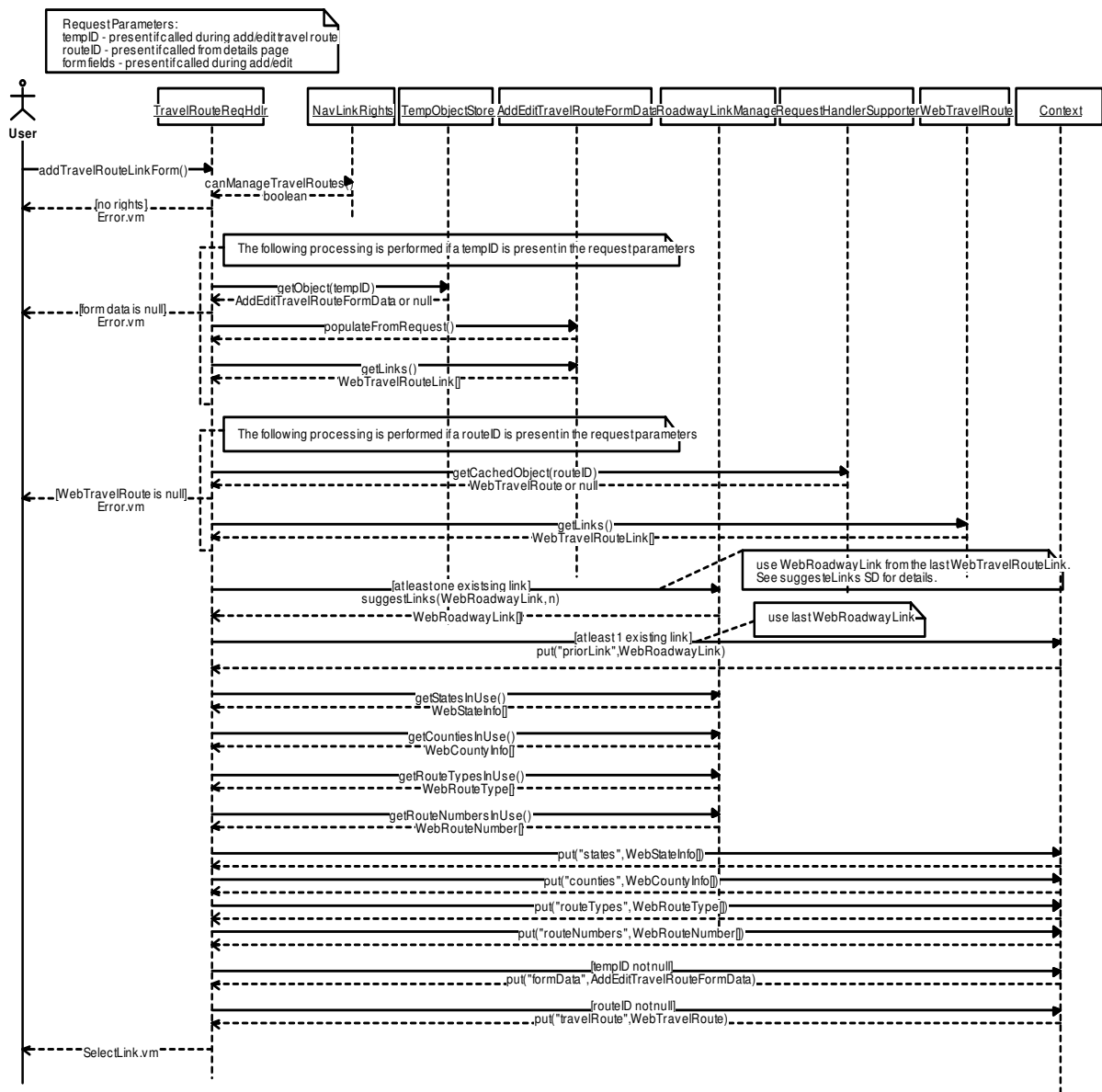


Figure 5-401. TravelRouteReqHdlr:addTravelRouteLinkForm (Sequence Diagram)

5.50.2.7 TravelRouteReqHdlr:findTravelRouteLinksJSON (Sequence Diagram)

This diagram shows the processing that is performed if the user chooses to search for links while selecting links to be added to a travel route. This request will be invoked via AJAX and will return JSON rather than HTML so that the search results will be displayed without requiring a page refresh. This will also save some bandwidth as JSON is very light weight compared to html.

The user rights are checked to make sure the user can manage travel routes, and if not a JSON response is returned to the browser so it can know the request failed and display an appropriate error message. If the user has rights to perform the operation, the request parameters that contain the search criteria are used to create a RoadwayLinkQuery. The RoadwayLinkManager is called to perform the query, and the list of all links that meet the query criteria are returned. The data from each link is loaded into a JSONArray and the array is stored inside an enclosing object (to make it easy for the Javascript in the browser to determine a valid result is returned). The object is then sent to the browser, and the method returns null to prevent the request handler from attempting to return its own response to the browser.

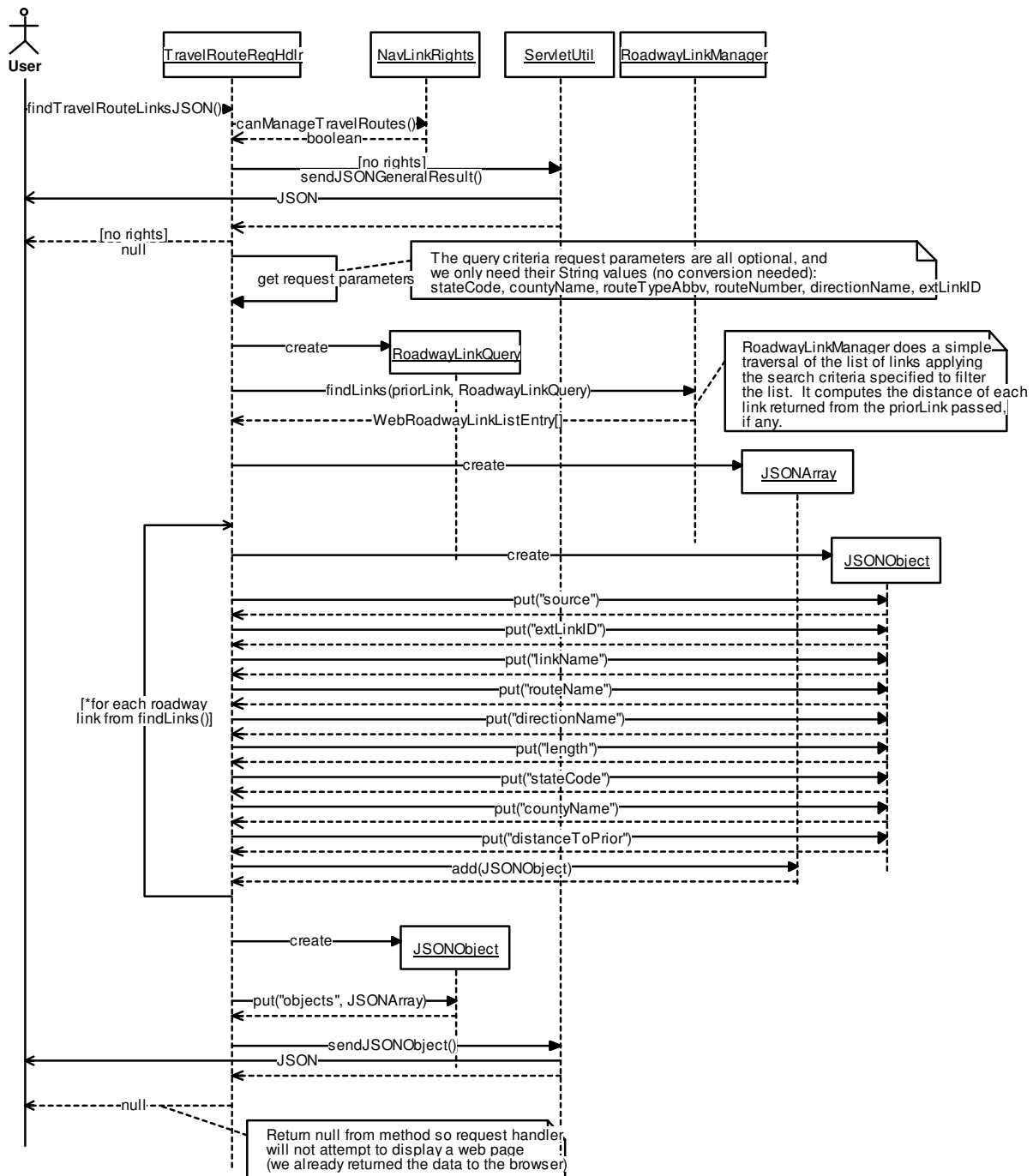


Figure 5-402. TravelRouteReqHdlr:findTravelRouteLinksJSON (Sequence Diagram)

5.50.2.8 TravelRouteReqHdlr:getHistoryBucketTimes (Sequence Diagram)

This diagram shows the processing performed to get the times that define the buckets used to display travel time and toll rate history in 5 minute increments. A Calendar is created and its time is set to the next 5 minute increment that occurs on the clock (00, 05, 10, etc.). The number of buckets is retrieved from the system profile, and then a loop is performed to store the bucket times in an array. The first array element will contain the first time computed (nearest 5 minute interval from now). Each successive bucket will contain a time 5 minutes earlier than the prior time.

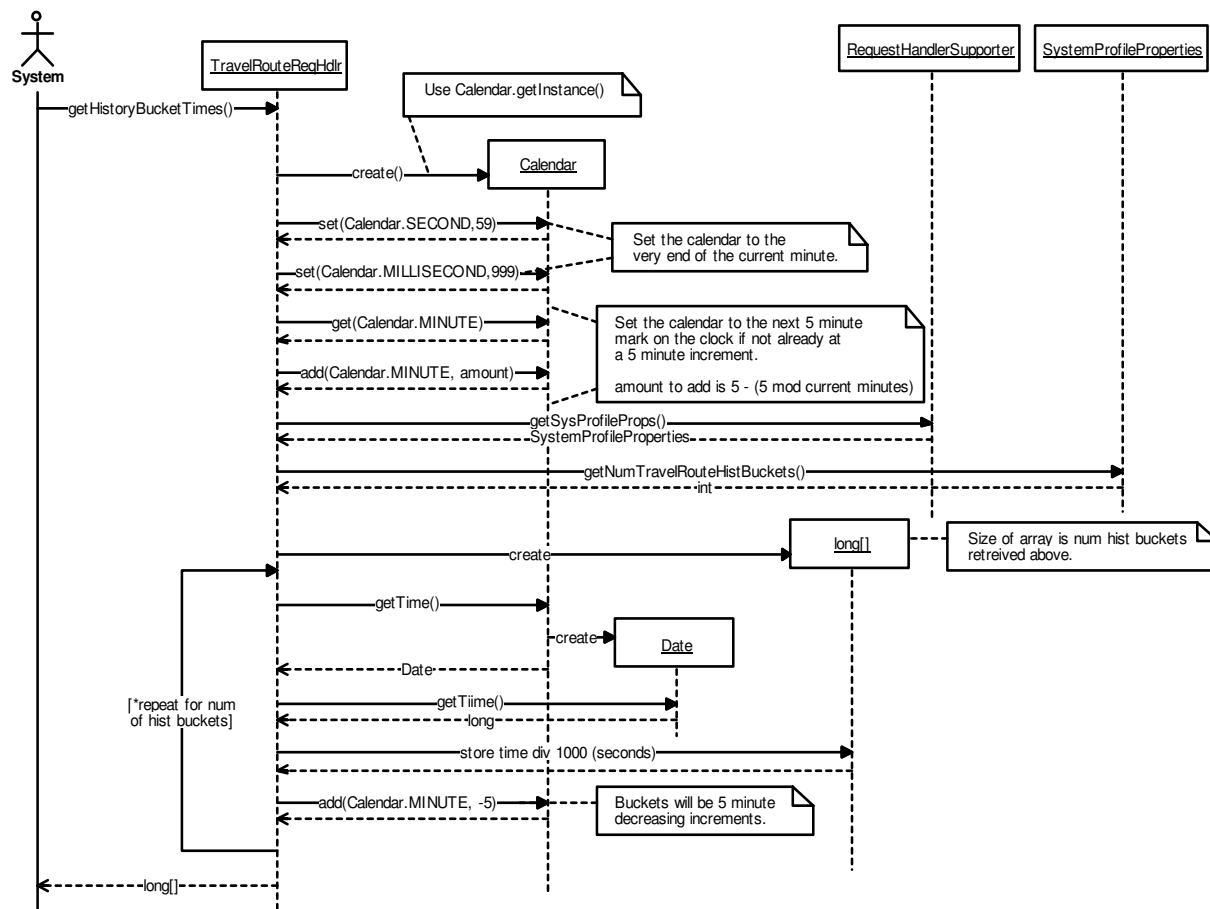


Figure 5-403. TravelRouteReqHdlr:getHistoryBucketTimes (Sequence Diagram)

5.50.2.9 TravelRouteReqHdr:moveTravelRouteLink (Sequence Diagram)

This diagram shows the processing that is performed when the user chooses to move a link up or down in the travel route's list of links. This can be done as part of the add/edit travel route process, or done stand alone from the travel route details page. When done from the add/edit process, a temp ID will be present in the request parameters and will be used to retrieve the form data object from the temp object store. The form data will be populated from the request to store the form field entries made by the user prior to invoking this action so that they will not be lost. The link is moved within the list of links in the form data, and the add/edit form is redisplayed to the user.

If the user has invoked this action from the travel route details page, a route ID will be present in the request parameters. The routeID will be used to retrieve the WebTravelRoute from the object cache, and the CORBA object reference for the travel route will be retrieved. The travel route is called to obtain the current configuration data, and the current configuration data is updated to move the specified link. This updated configuration data is passed back to the travel route in a call to setConfig(), and the WebTravelRoute object is updated with this new configuration data. The user is then redirected back to the travel route details page.

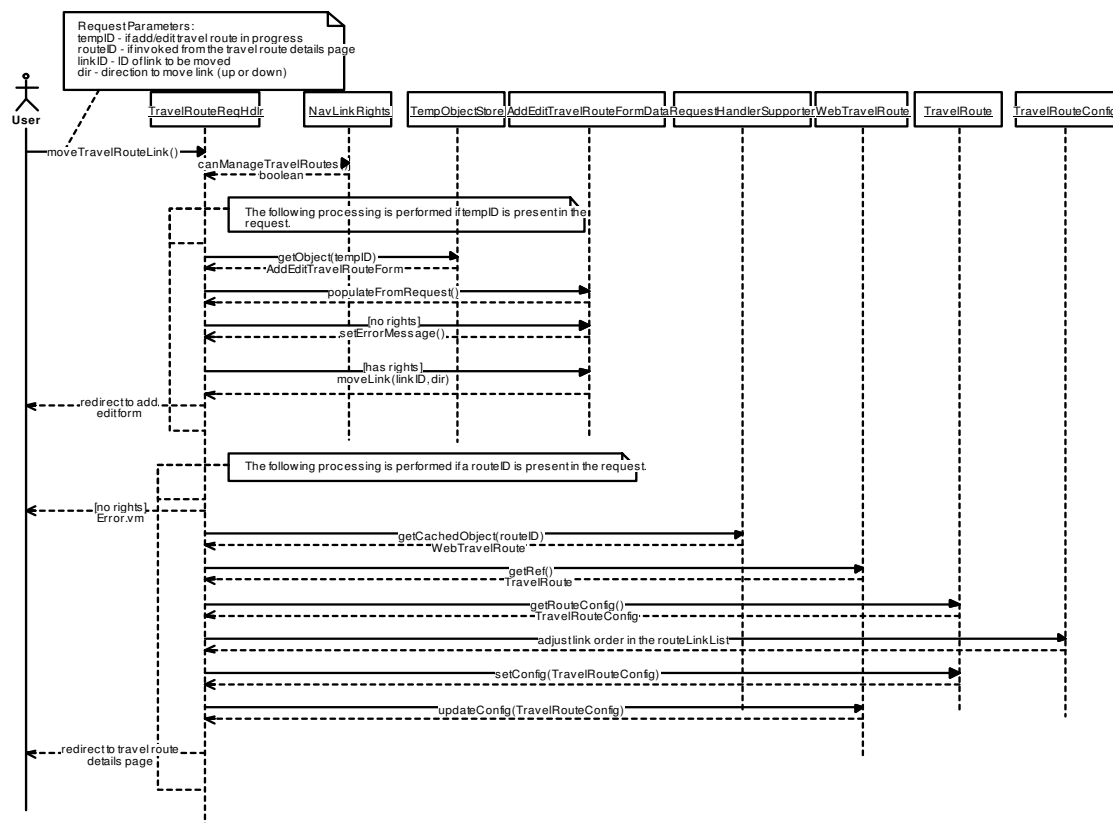


Figure 5-404. TravelRouteReqHdr:moveTravelRouteLink (Sequence Diagram)

5.50.2.10 TravelRouteReqHdr:removeTollRateSource (Sequence Diagram)

This diagram shows the processing that is performed when the user chooses to remove the toll rate source for a travel route. This can be done as part of the add/edit travel route process, or done stand alone from the travel route details page. When done from the add/edit process, a temp ID will be present in the request parameters and will be used to retrieve the form data object from the temp object store. The form data will be populated from the request to store the form field entries made by the user prior to invoking this action so that they will not be lost. The toll rate route is removed from the form data, and the add/edit form is redisplayed to the user.

If the user has invoked this action from the travel route details page, a route ID will be present in the request parameters. The routeID will be used to retrieve the WebTravelRoute from the object cache, and the CORBA object reference for the travel route will be retrieved. The travel route is called to obtain the current configuration data, and the current configuration data is updated to remove the toll rate configuration. This updated configuration data is passed back to the travel route in a call to setConfig(), and the WebTravelRoute object is updated with this new configuration data. The user is then redirected back to the travel route details page.

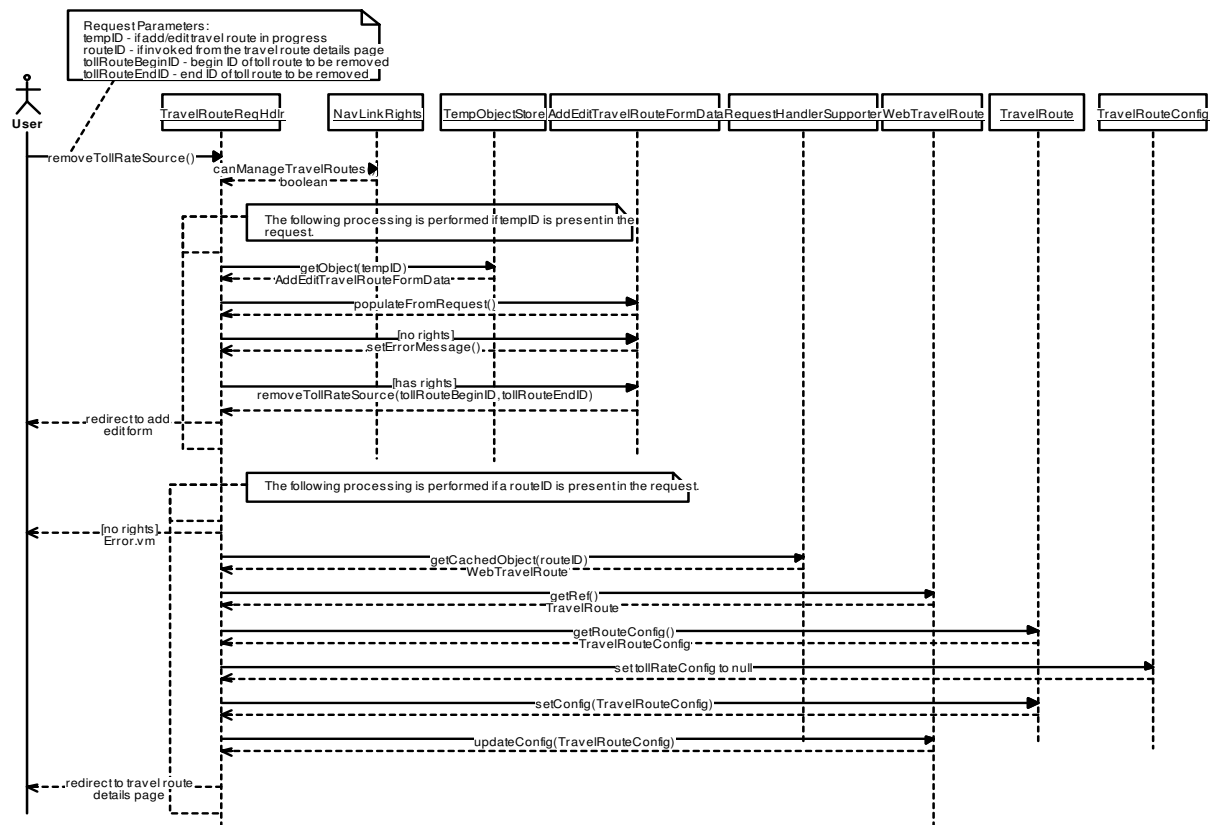


Figure 5-405. TravelRouteReqHdr:removeTollRateSource (Sequence Diagram)

5.50.2.11 TravelRouteReqHdlr:removeTravelRoute (Sequence Diagram)

This diagram shows the processing that is performed when the user chooses to remove a travel route from the system. The user will have already confirmed their intent prior to this processing being invoked. The user's rights are checked and an error is returned if they do not have permission to manage travel routes. The cached travel route object is retrieved from the object cache and its CORBA object reference is obtained and called to remove the travel route from the system. The data model's objectRemoved method is called to remove the object from the GUI cache.

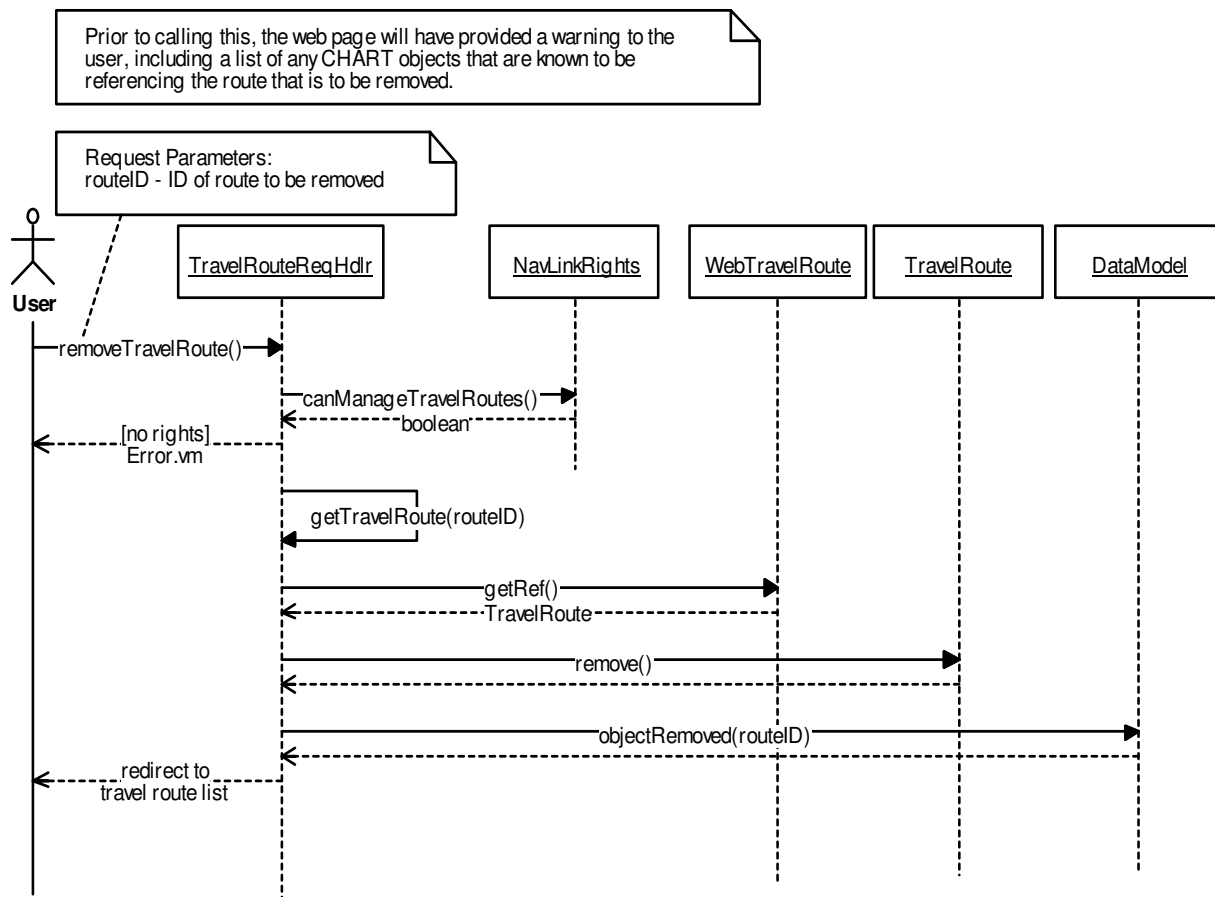


Figure 5-406. TravelRouteReqHdlr:removeTravelRoute (Sequence Diagram)

5.50.2.12 TravelRouteReqHdr:removeTravelRouteLink (Sequence Diagram)

This diagram shows the processing that is performed when the user chooses to remove a link from a travel route. This can be done as part of the add/edit travel route process, or done stand alone from the travel route details page. When done from the add/edit process, a temp ID will be present in the request parameters and will be used to retrieve the form data object from the temp object store. The form data will be populated from the request to store the form field entries made by the user prior to invoking this action so that they will not be lost. The link is removed from the form data, and the add/edit form is redisplayed to the user.

If the user has invoked this action from the travel route details page, a route ID will be present in the request parameters. The routeID will be used to retrieve the WebTravelRoute from the object cache, and the CORBA object reference for the travel route will be retrieved. The travel route is called to obtain the current configuration data, and the current configuration data is updated to remove the specified link. This updated configuration data is passed back to the travel route in a call to setConfig(), and the WebTravelRoute object is updated with this new configuration data. The user is then redirected back to the travel route details page.

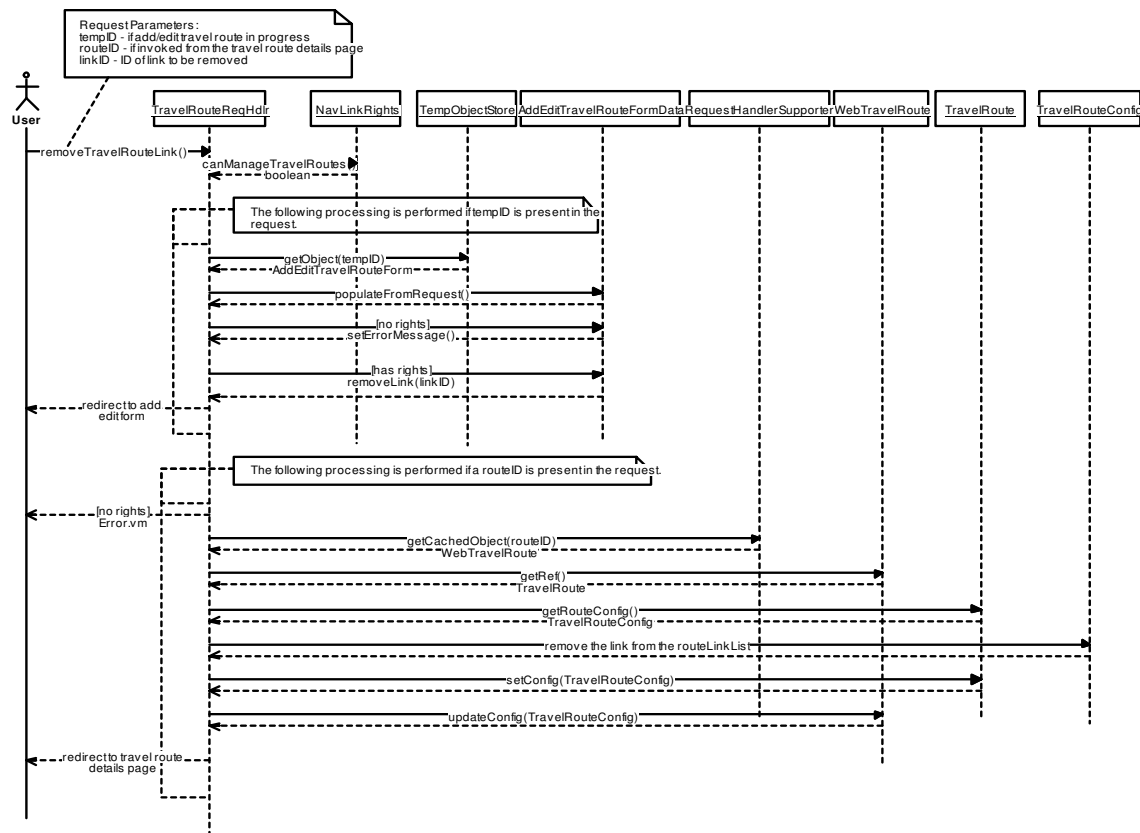


Figure 5-407. TravelRouteReqHdr:removeTravelRouteLink (Sequence Diagram)

5.50.2.13 TravelRouteReqHdlr:setTollRateSource (Sequence Diagram)

This diagram shows the processing that takes place when the select toll rate source form is submitted. The user's rights are checked and an error message is shown if they don't have the manage travel routes right. The request parameters are read and a TollRateRouteSpec is created using the start ID, end ID, external system id, and description passed in the request parameters.

If a tempID is in the request parameters, this operation is being done as part of the add/edit travel route operation and the tempID is used to obtain the AddEditTravelRouteFormData object from the temp object store. The TollRateRouteSpec is stored in the form data object, and the request is redirected to the add/edit travel route form where the newly added toll rate source will appear.

If a routeID is in the request parameters, this operation is being done stand alone from the travel route details page. The WebTravelRoute is found in the GUI cache using the routeID and the TravelRoute CORBA object reference is obtained. The TravelRoute is called to get its current configuration, and the TollRateRouteSpec is used to replace the existing TollRateRouteSpec in the TravelRouteConfig. This may require a TollRateConfig to be created, as it is an optional element of the TravelRouteConfig. The TravelRouteConfig is then used in the call to the TravelRoute's setTollRateConfig method, and the cached WebTravelRoute's TravelRouteConfig is also updated. The request is then redirected to the travel route details page.

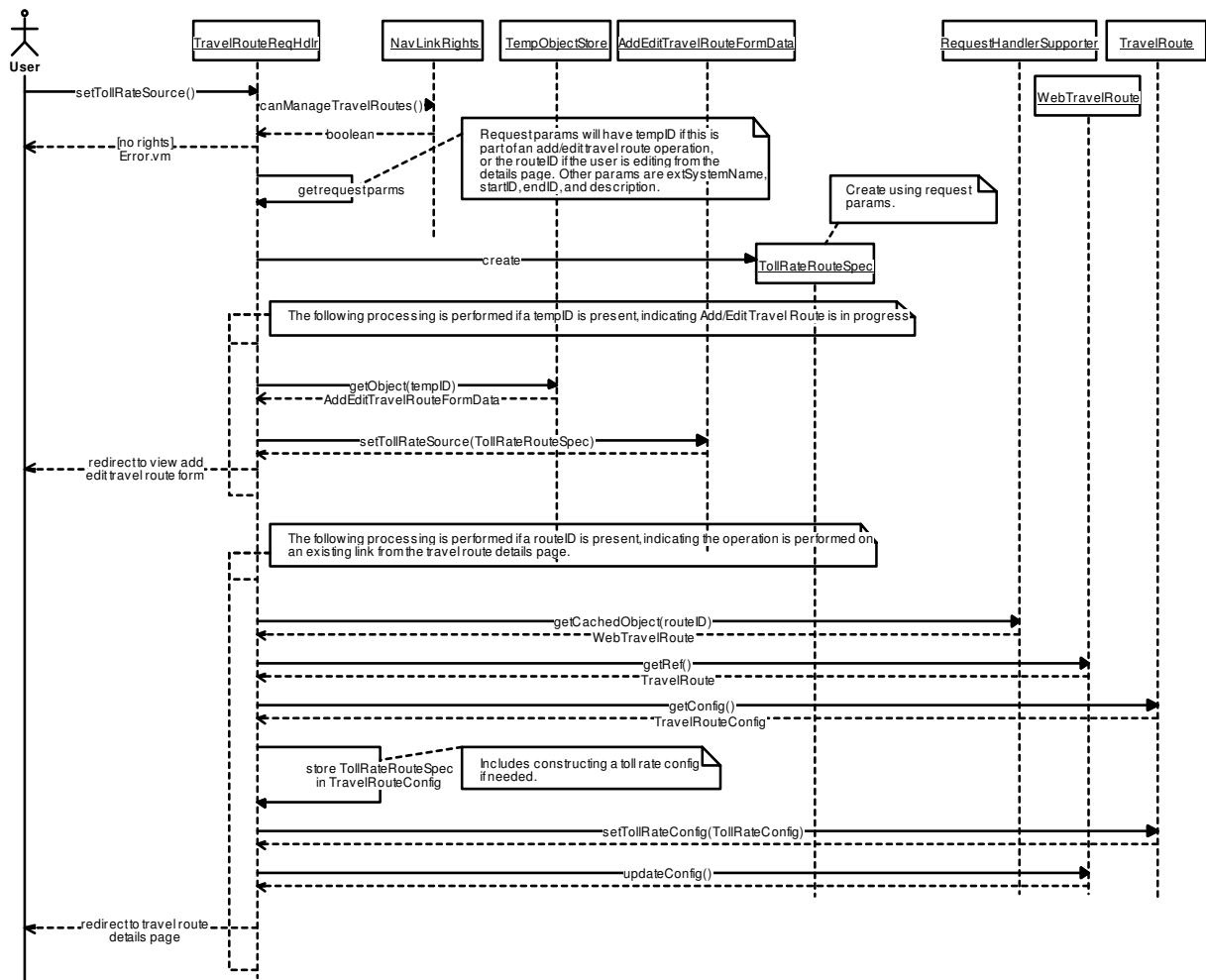


Figure 5-408. TravelRouteReqHdr:setTollRateSource (Sequence Diagram)

5.50.2.14 TravelRouteReqHdlr:setTollRateSourceForm (Sequence Diagram)

This diagram shows the processing that is performed when a user wishes to set the toll rate source for a travel route. This action can be performed as part of the add/edit travel route operation, or as a stand alone operation from the travel route details page. When done as part add/edit travel route, a tempID will be present the request parameters and it will be used to obtain the AddEditTravelRouteFormData from the temporary object store. The form data is updated to save the data fields from the add/edit travel route form so the user's current entries will not be lost. The temp ID and form data are put in the context.

If this action is performed from the travel route details page, a route ID will be present in the request parameters and will be used to find the WebTravelRoute in the GUI cache. The route ID and WebTravelRoute are placed in the context.

The TollRateRouteManager is called to obtain the list of known toll rate routes and this list of toll rate routes is placed in the context. The SelectTollRateSource form is then shown to the user.

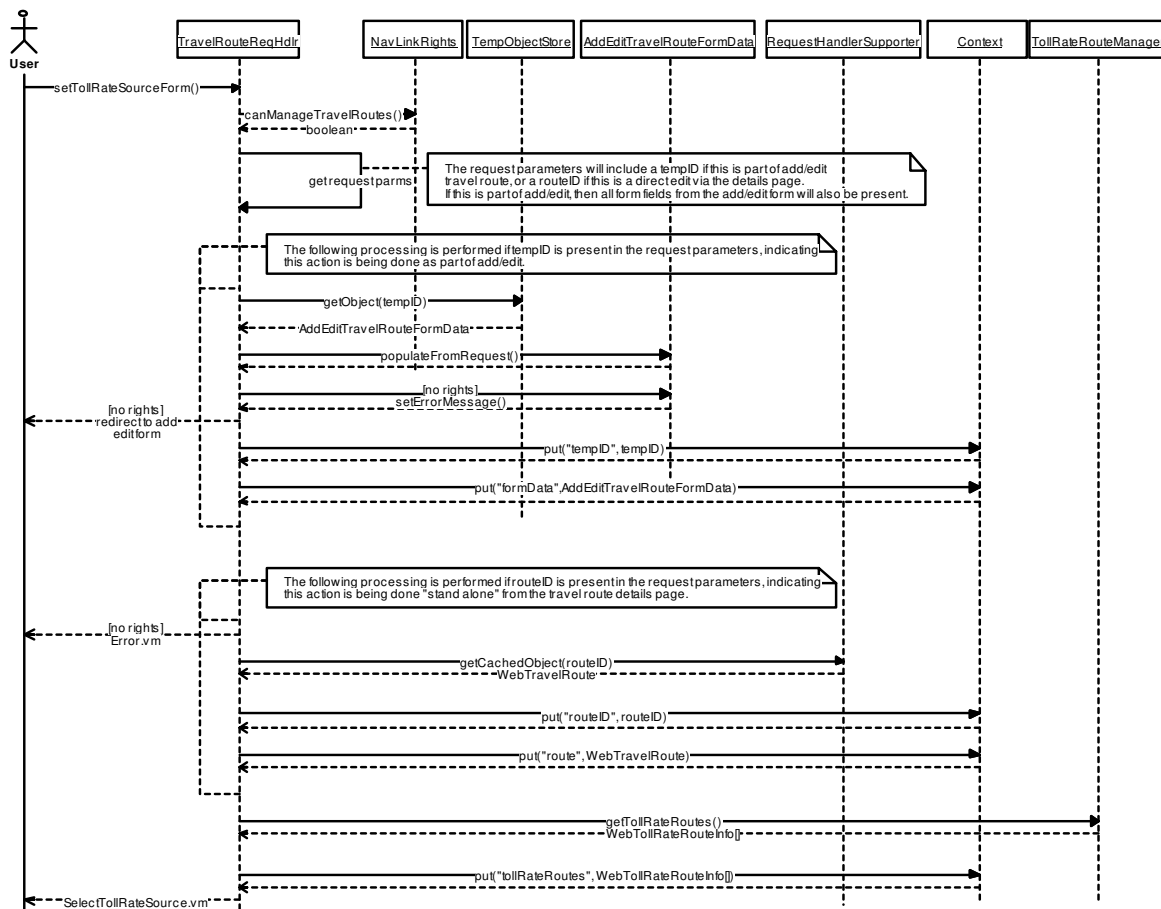


Figure 5-409. TravelRouteReqHdlr:setTollRateSourceForm (Sequence Diagram)

5.50.2.15 TravelRouteReqHdr:setTravelRouteLinkSettings (Sequence Diagram)

This diagram shows the processing that takes place when the user submits the form used to set the settings for a link that's included in a travel route. The form may be used during an add/edit operation on a travel route, or directly from the travel route's details page. When used as part of an add/edit operation, the tempID parameter will be present in the request and is used to retrieve the form data from the temp object store. The WebTravelRouteLink is obtained from the form data and its settings are changed using the request parameters (percent and minimum quality). The request is then redirected to display the add/edit travel route form.

When this form is used from the travel route details page, the route ID will be present in the request and is used to find the WebTravelRoute in the GUI cache. The CORBA object reference for the TravelRoute is obtained and called to get the latest configuration data for the travel route. The settings for the link within the travel time config portion of the travel route config are changed as specified in the request parameters, and the TravelRoute is called to set its travel time settings. The WebTravelRouteLink is also updated so the GUI cache will have this latest data without having to wait on a CORBA event. The request is redirected to display the travel route details page.

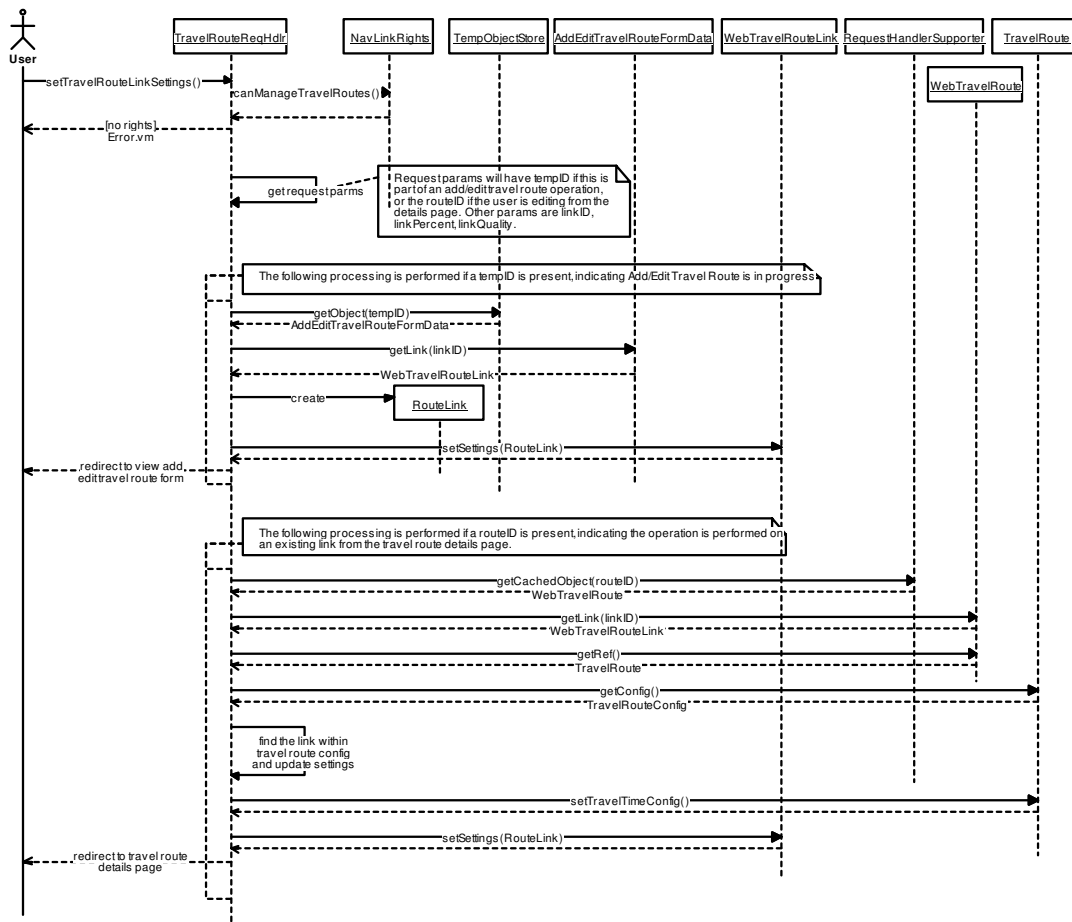


Figure 5-410. TravelRouteReqHdr:setTravelRouteLinkSettings (Sequence Diagram)

5.50.2.16 TravelRouteReqHdr:setTravelRouteLinkSettingsForm (Sequence Diagram)

This diagram shows the processing that is performed when the user chooses to set the settings for a link included in a travel route (the percent of link included and minimum quality). This form can be used as part of the add/edit travel route operation, or as a stand alone operation invoked from the travel route details page. When used as part of the add/edit travel route operation, the tempID parameter will be present in the request and will be used to obtain the form data object from the temp object store. The form data's populateFromRequest method will be called to save the field data from the add/edit travel route form so the user doesn't lose their work while they work on this auxillary form. The link whose settings are being edited is obtained from the form data object, and the tempID and formData are placed in the context. The form data is put in the context to allow the form show the name of the travel route, which may only exist in the form data if it is an add operation (it hasn't been added to the system yet).

If this form was invoked from the travel route details page, the route ID will be present in the form data and will be used to retrieve the WebTravelRoute from the GUI cache. The WebTravelRouteLink is obtained from the WebTravelRoute, and the route ID and WebTravelRoute are placed in the context. The WebTravelRoute is put in the context so its name can be shown on the form.

Lastly, the WebTravelRouteLink whose settings are being changed is placed in the context, and the form is returned.

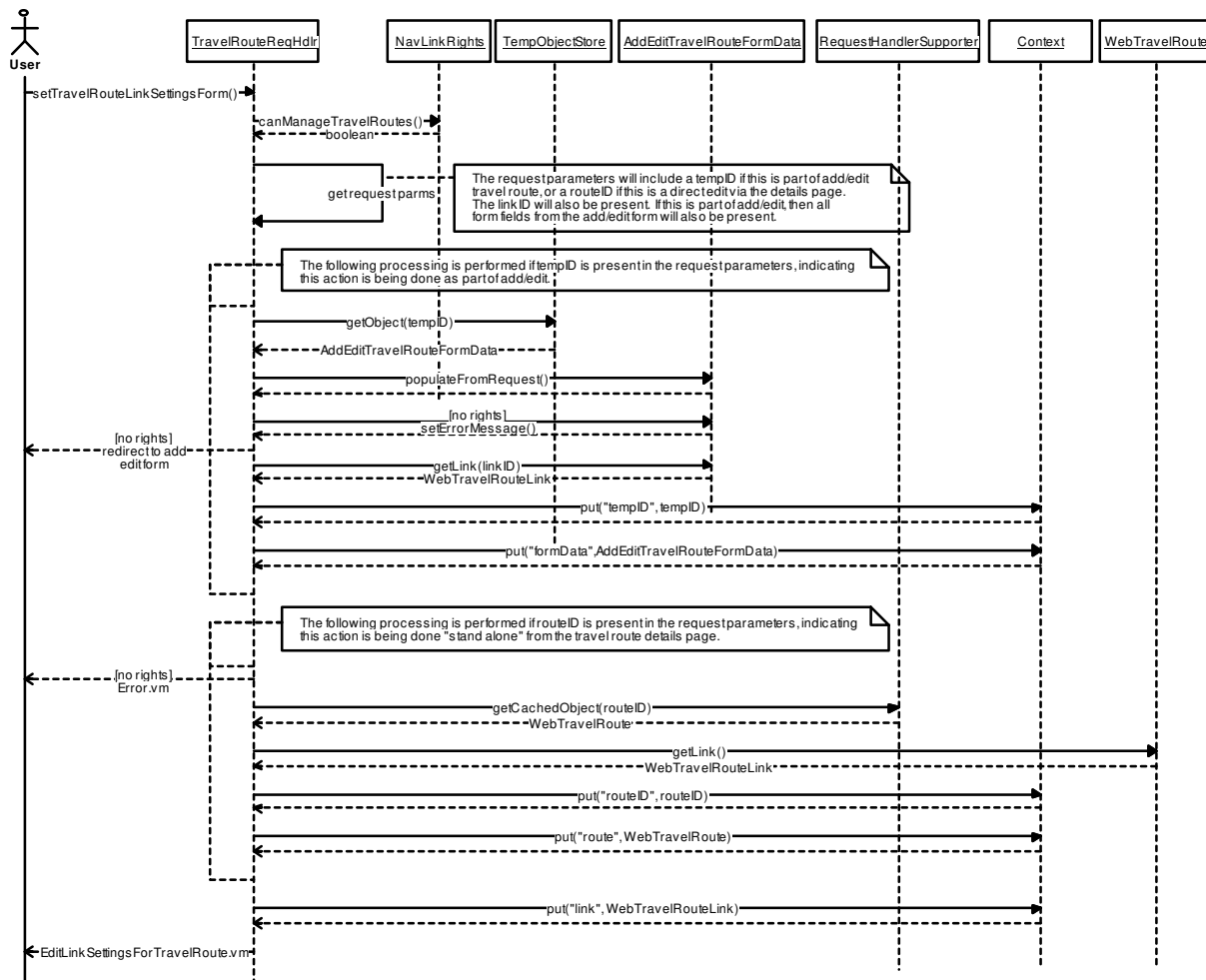


Figure 5-411. TravelRouteReqHdlr:setTravelRouteLinkSettingsForm (Sequence Diagram)

5.50.2.17 TravelRouteReqHdlr:viewTravelRouteDetails (Sequence Diagram)

This diagram shows the processing performed when the user requests to view the details page for a travel route. The user's rights are checked and an error is returned if they do not have the right to view travel routes. The travel route requested is retrieved from the object cache and placed in the velocity context. The times used to specify the buckets used to show travel time and toll rate history are computed (see `getTravelRouteBucketTimes` diagram) and placed in the context. The `WebTravelRoute` is called to get the list of objects using it, and the list is placed in the context.

The `TravelRouteDetails.vm` template is returned to display data from the travel route in the details page. When the page displays history, it can use the bucket travel times included in the context to pass to `HistoryList.toBucketArray()` to get the historical values in the proper "buckets" that match the bucket times. By using the same bucket times for each link's travel time history, the route's travel time history, and the route's toll rate history, the page will display the data in consistent buckets.

When displaying travel times for the route (current or history), the details page will check the status flag associated with each route travel time and display the travel time with an indicator for travel times that are not valid due to expiration, link data quality, exceeded max, etc. The same is true for the current toll rate and toll rate history. There is no such status for link level travel time data aside from quality.

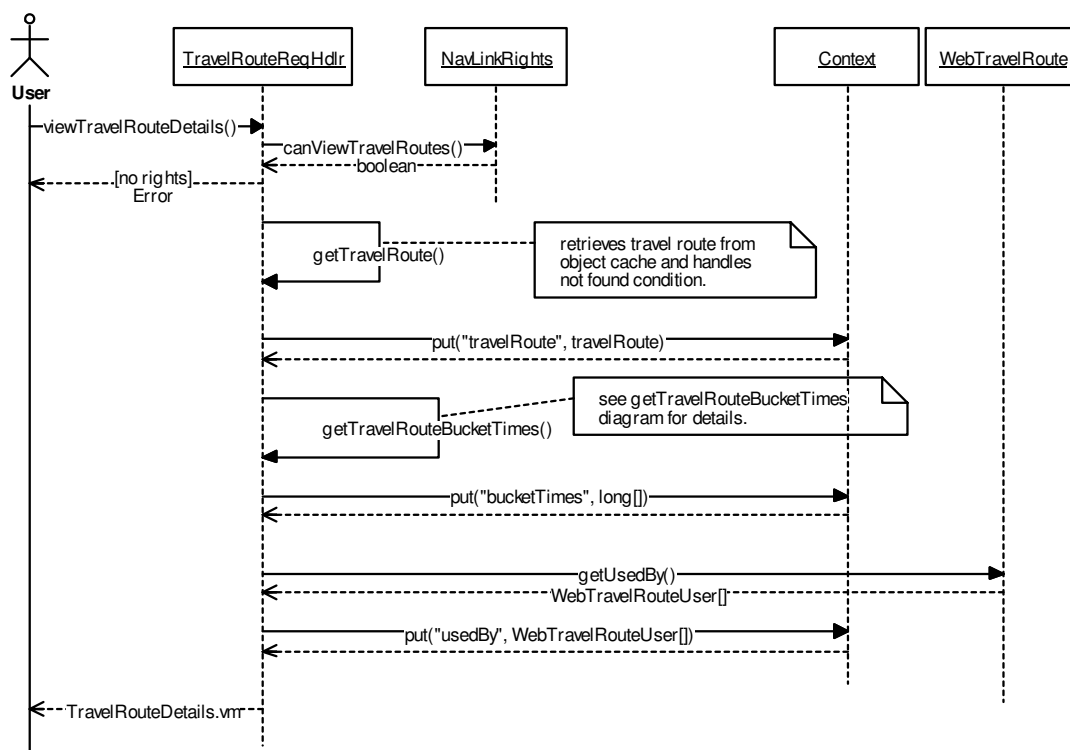


Figure 5-412. TravelRouteReqHdlr:viewTravelRouteDetails (Sequence Diagram)

5.50.2.18 TravelRouteReqHdlr:viewTravelRouteLinkDetails (Sequence Diagram)

This diagram shows the processing that is done when the user requests to view the details page for a roadway link. The user's rights are checked to make sure they are permitted to view travel routes, and if not an error is returned. The WebTravelRoute is retrieved from the object cache and its `getLink()` method is called to retrieve the WebTravelRouteLink. The link is placed in the velocity Context and the LinkDetails template is returned to display the details from the link on the link details page.

Note that unlike the processing for the travel route details page, bucket times are NOT obtained and placed in the context. This is because on the link details page we will show the actual time from each history record - we don't need to place the times in consistent buckets because we are only showing the travel time history data for a single link.

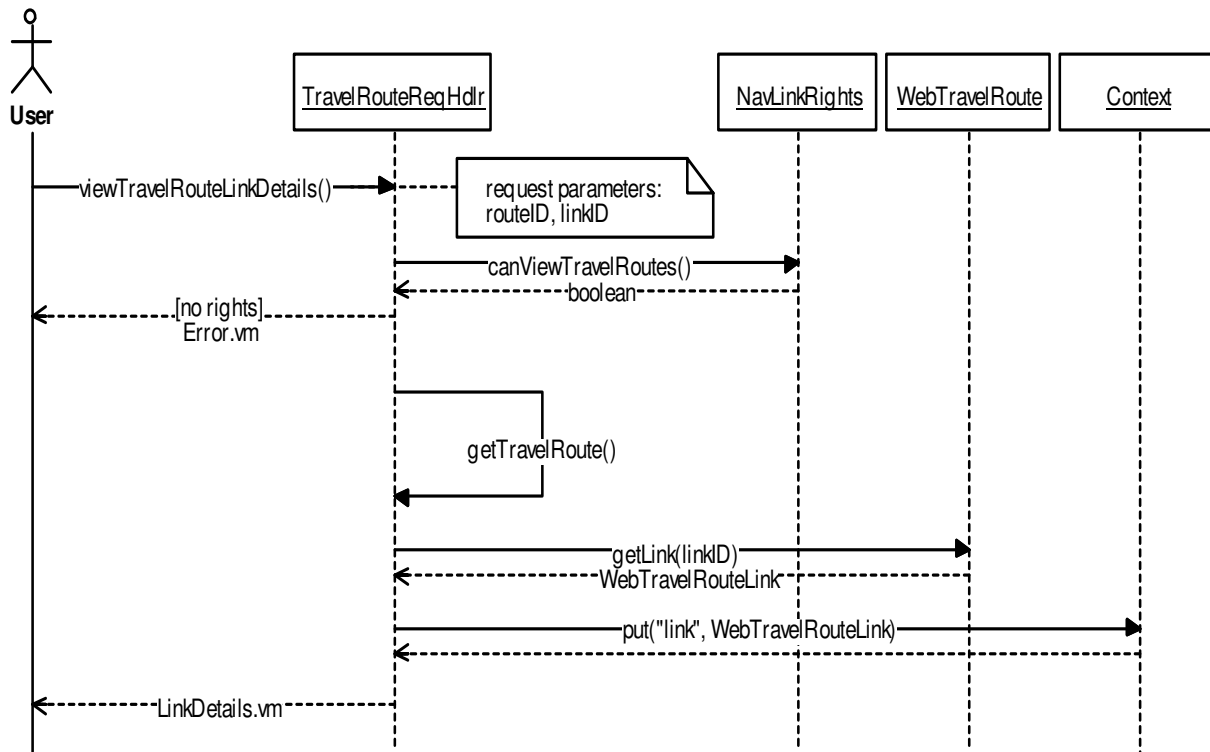


Figure 5-413. TravelRouteReqHdlr:viewTravelRouteLinkDetails (Sequence Diagram)

5.50.2.19 TravelRouteReqHdlr:viewTravelRoutes (Sequence Diagram)

This diagram shows the processing performed when a user chooses to view the travel routes defined in the system. Note that much of the processing is handled by the DynListReqHdlrDelegate class, a mature existing class that is not being modified for use with travel routes. For this reason, much of its processing is summarized with text rather than showing the details. The points where it interacts with Travel Route specific classes (namely the TravelRouteDynListSupporter) are shown in their proper context.

After receiving the request to view travel routes, the user's rights are checked to make sure they have rights to view travel routes. If not, they will be shown an error page. If the user has proper rights, the processing is passed to the DynListReqHdlrDelegate. It checks to see if the list is already displayed, and if not it calls the TravelRouteDynListSupporter to create a DynList (see the createDynList diagram for details). After creating the dynamic list, it redirects the browser to the view list request, which will result in this exact same processing being invoked again, except the list will now exist.

If the list exists, it is touched to make sure it doesn't expire from the temp object store. The subjects for the list are then retrieved (see the getDynListSubjects diagram for details). The subjects are placed into the dynamic list, and the delegate clears any filters that need to be cleared (as specified via request parameters). The delegate then loads the context, and returns to the TravelRouteReqHdlr. The request handler checks to see if the delegate encountered an error, and if it did, it returns the error message supplied. Otherwise, the request handler retrieves all WebTravelRouteUser objects from the GUI cache and calls each one to find the travel routes it is currently using. This information is stored in a hashtable so that the travel route list can easily retrieve a list of objects using it for the "used by" column of the travel route list. After constructing this hashtable and placing it in the context, the travel route list will be shown to the user.

Note that the travel time trend shown in the route list is available directly from the travel route's status (computed by the server), and the list of recent travel time history shown when the user hovers over the trend is available directly from the HistoryList in the WebTravelRouteStatus.

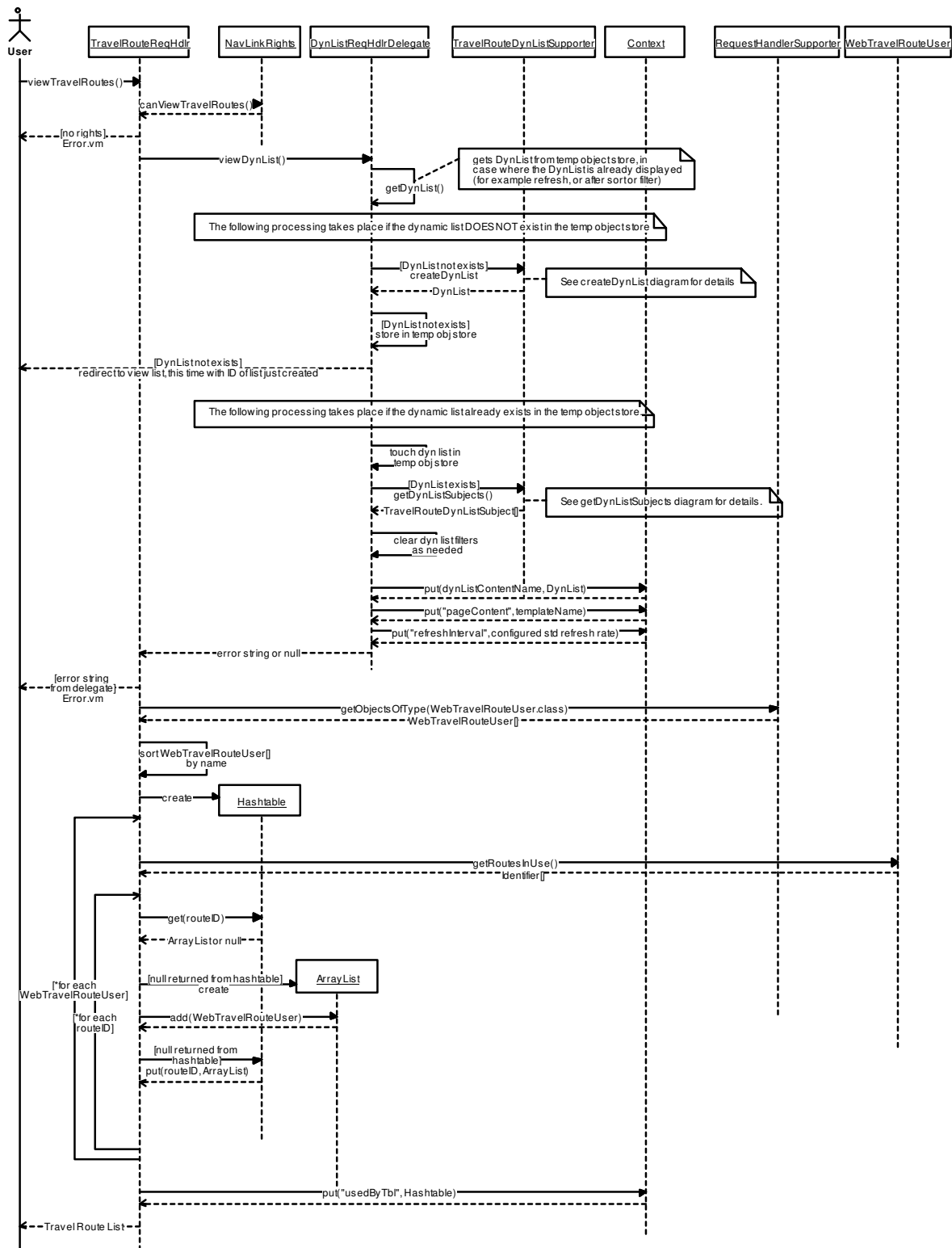


Figure 5-414. TravelRouteReqHdlr:viewTravelRoutes (Sequence Diagram)

5.51 Chartlite.servlet.externalsystemmgmt

5.51.1 ClassFiles

5.51.1.1 GUIExternalSystemServletClasses (Class Diagram)

This diagram shows classes used by the GUI (servlet) to support requests related to the external system.

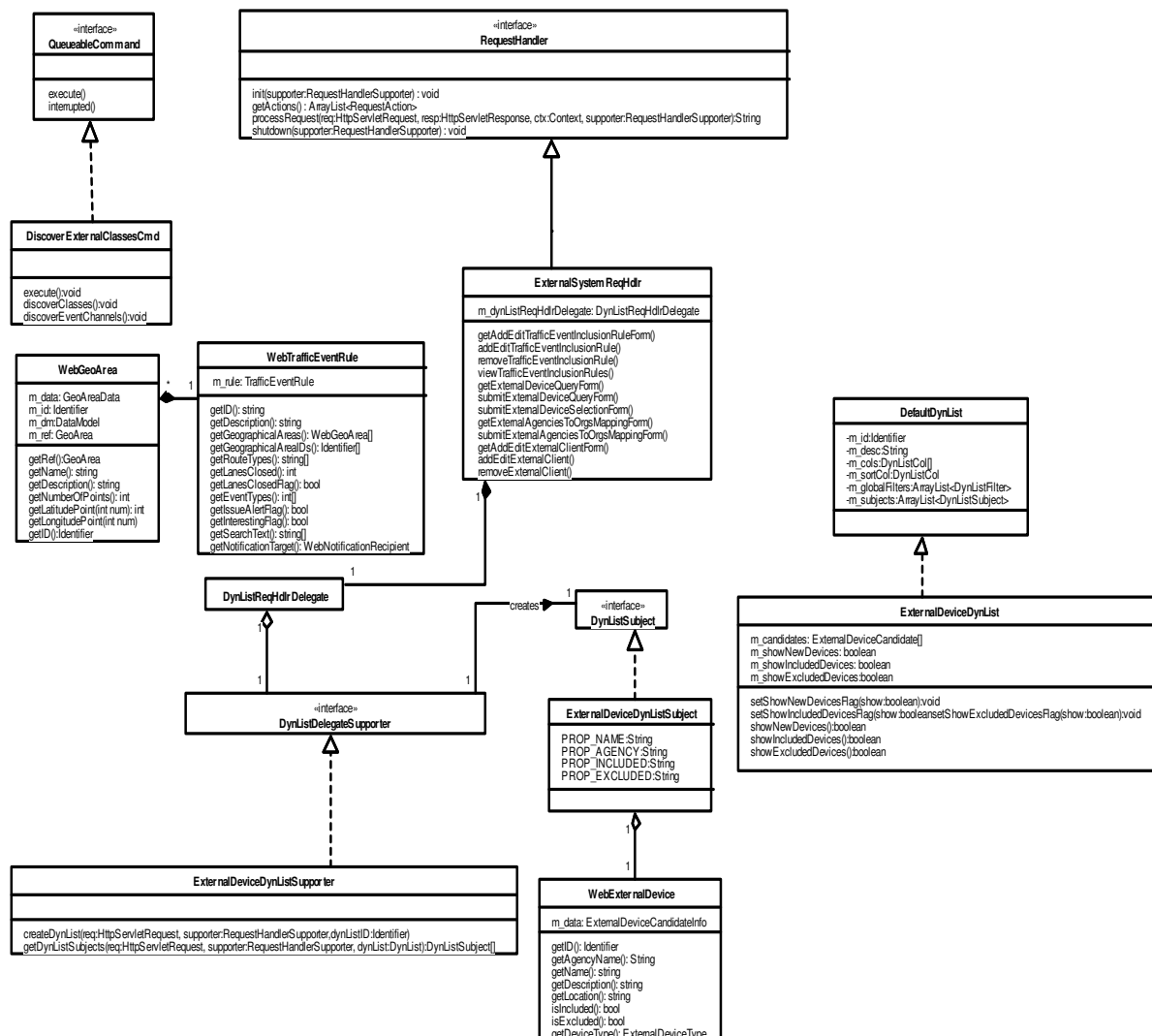


Figure 5-415. GUIExternalSystemServletClasses (Class Diagram)

5.51.1.1.1 DefaultDynList (Class)

This class provides a default implementation of the DynList interface. It supports a collection of columns, a collection of global filters, and a collection of subjects. Filters in this list are treated additively - that is, a subject must pass all filters to be displayed.

5.51.1.1.2 DiscoverExternalClassesCmd (Class)

This class is called to periodically discover classes related to the external system from the trading service and External Interface module. This includes traffic event rules, and external connections

5.51.1.1.3 DynListDelegateSupporter (Class)

This interface contains functionality to support the DynListReqHdlrDelegate

5.51.1.1.4 DynListReqHdlrDelegate (Class)

This class helps request handlers support dynamic lists. Requests to view, sort, or filter dynamic lists can be passed from a request handler to this class, provided the URL used for the requests contain parameters required by this class, such as the id of the list, the property name, and/or the filter value.

5.51.1.1.5 DynListSubject (Class)

This interface is implemented by classes that wish to be capable of being displayed in a dynamic list.

5.51.1.1.6 ExternalDeviceDynList (Class)

This class implements the dynlist interface for external devices in dynamic lists.

5.51.1.1.7 ExternalDeviceDynListSubject (Class)

This class implements the DynListSubject interface and contains fields for displaying external devices in dynamic lists.

5.51.1.1.8 ExternalDeviceDynListSupporter (Class)

This class implements the DynListDelegateSupporter for creating dynamic lists of external devices.

5.51.1.1.9 ExternalSystemReqHdlr (Class)

This class contains request methods related to external systems.

5.51.1.1.10 QueueableCommand (Class)

A QueueableCommand is an interface used to represent a command that can be placed on a CommandQueue for asynchronous execution. Derived classes implement the execute

method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a CommandQueue. The CommandQueue driver calls the execute method to execute a command in the queue and a call to the interrupted method is made when a CommandQueue is shut down.

5.51.1.1.11RequestHandler (Class)

This interface specifies methods that are to be implemented by classes that are used to process requests.

5.51.1.1.12WebExternalDevice (Class)

This class wraps an External Device for display in the GUI.

5.51.1.1.13WebGeoArea (Class)

This class wraps a GeoArea object to provide methods for accessing the data from a velocity template.

5.51.1.1.14WebTrafficEventRule (Class)

This class wraps a TrafficEventRule for display in the GUI

5.51.2 Sequence Diagrams

5.51.2.1 ExternalDeviceDynListSupporter:createDynList (Sequence Diagram)

This diagram shows the processing used to create an external object dynamic list. Each column of the list is created, along with comparators and filters used for each column. Those shown that do not create a filter are not filterable. Those columns that do not show construction of a comparator use the default text comparator that will be created by the DefaultDynListCol class. After creating the columns, they are passed to the constructor of a DefaultDynList, and the DefaultDynList is returned.

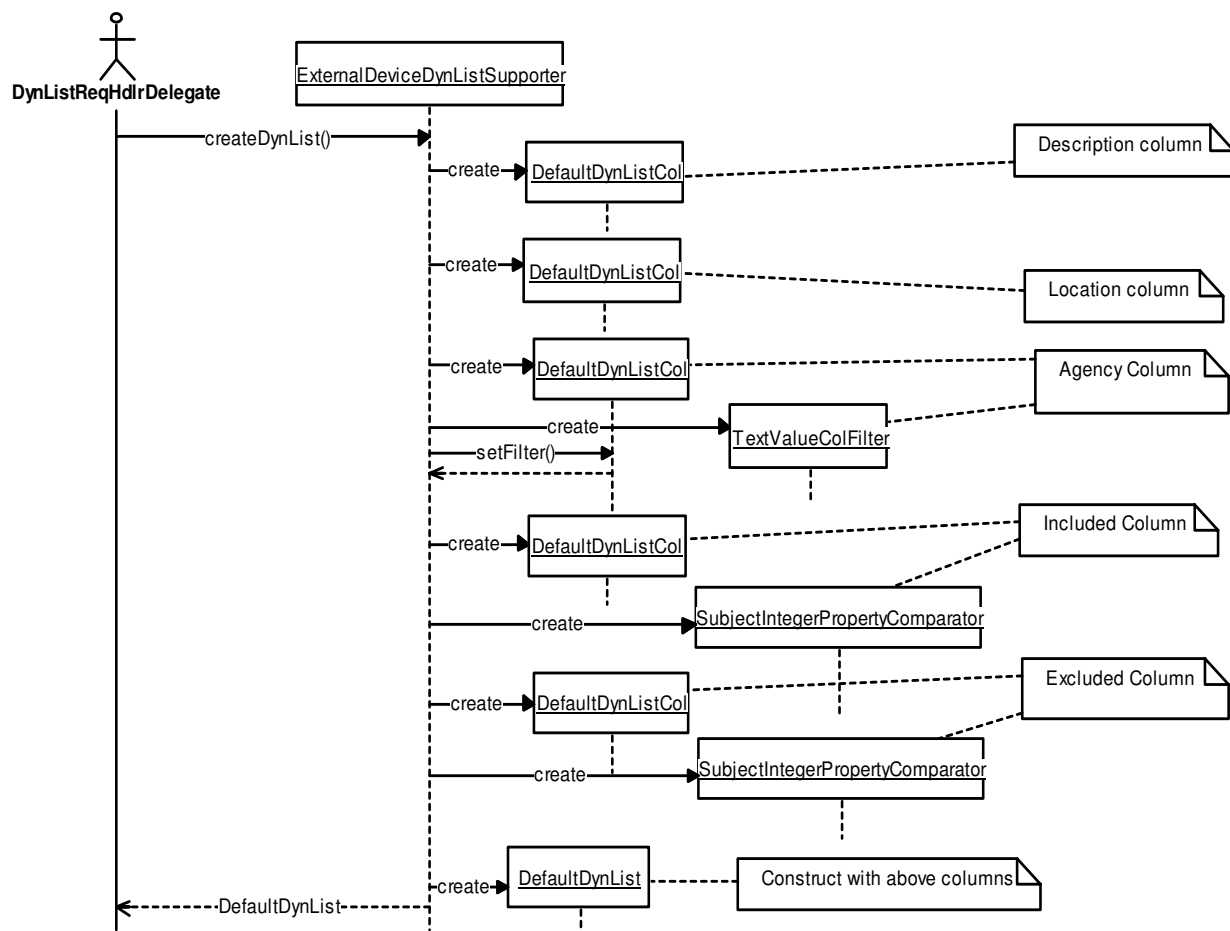


Figure 5-416. ExternalDeviceDynListSupporter:createDynList (Sequence Diagram)

5.51.2.2 ExternalDeviceDynListSupporter:getDynListSubjects (Sequence Diagram)

This diagram shows the processing performed when the ExternalDeviceDynListSupporter is called to get the dyn list subjects. All queried candidates are evaluated by a state filter, wrapped with a ExternalDeviceDynListSubject. A list of filtered subjects is returned.

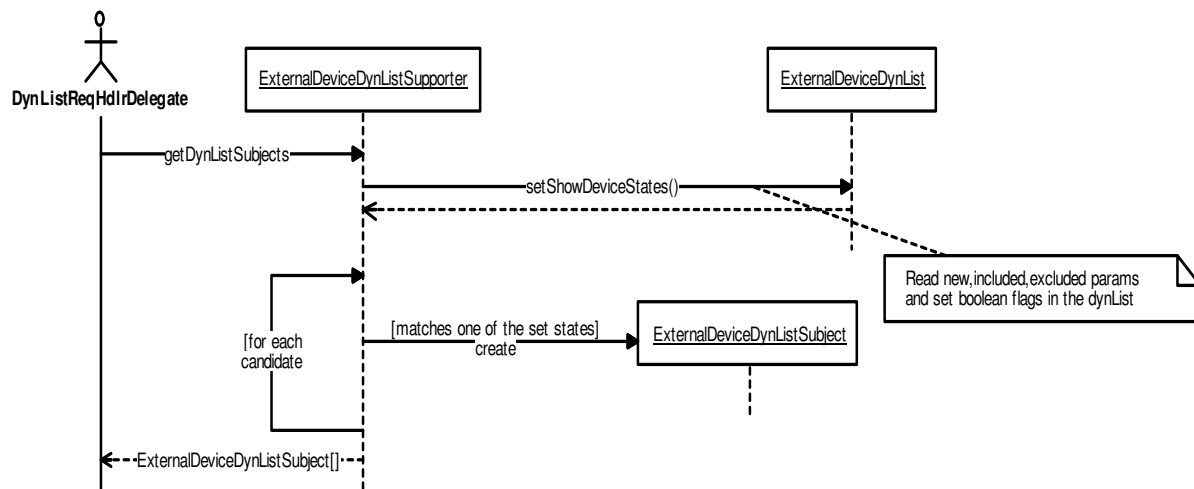


Figure 5-417. ExternalDeviceDynListSupporter:getDynListSubjects (Sequence Diagram)

5.51.2.3 ExternalSystemReqHdlr:addEditExternalClient (Sequence Diagram)

This diagram shows the processing that occurs when an administrator adds or edits an external client.

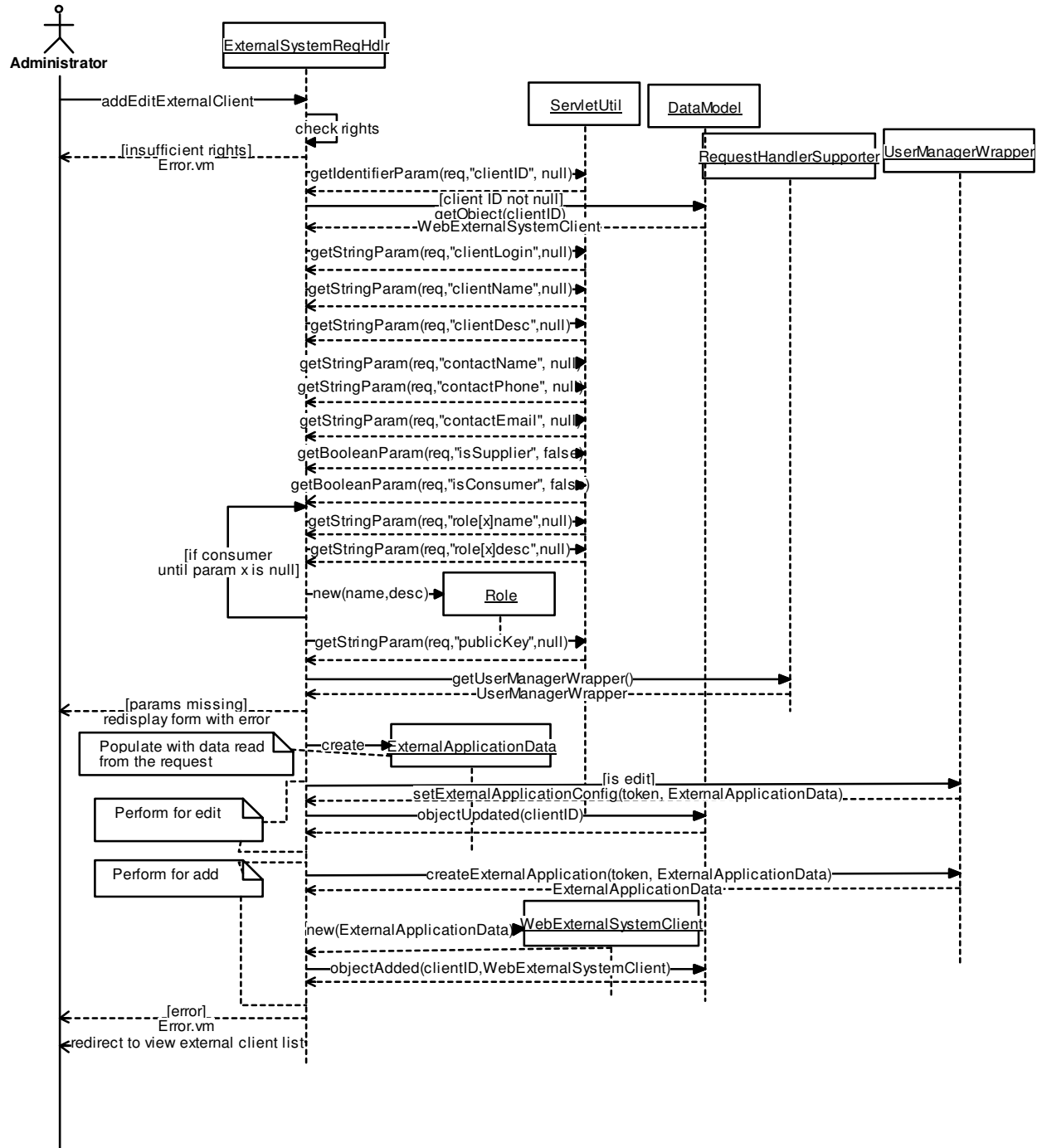


Figure 5-418. ExternalSystemReqHdlr:addEditExternalClient (Sequence Diagram)

5.51.2.4 ExternalSystemReqHdlr:addEditTrafficEventInclusionRule (Sequence Diagram)

This diagram shows the processing that occurs when an administrator has chosen to add/edit a new external event inclusion rule. The administrator selects/updates the values from the add/edit external event rule form and submits the rule form

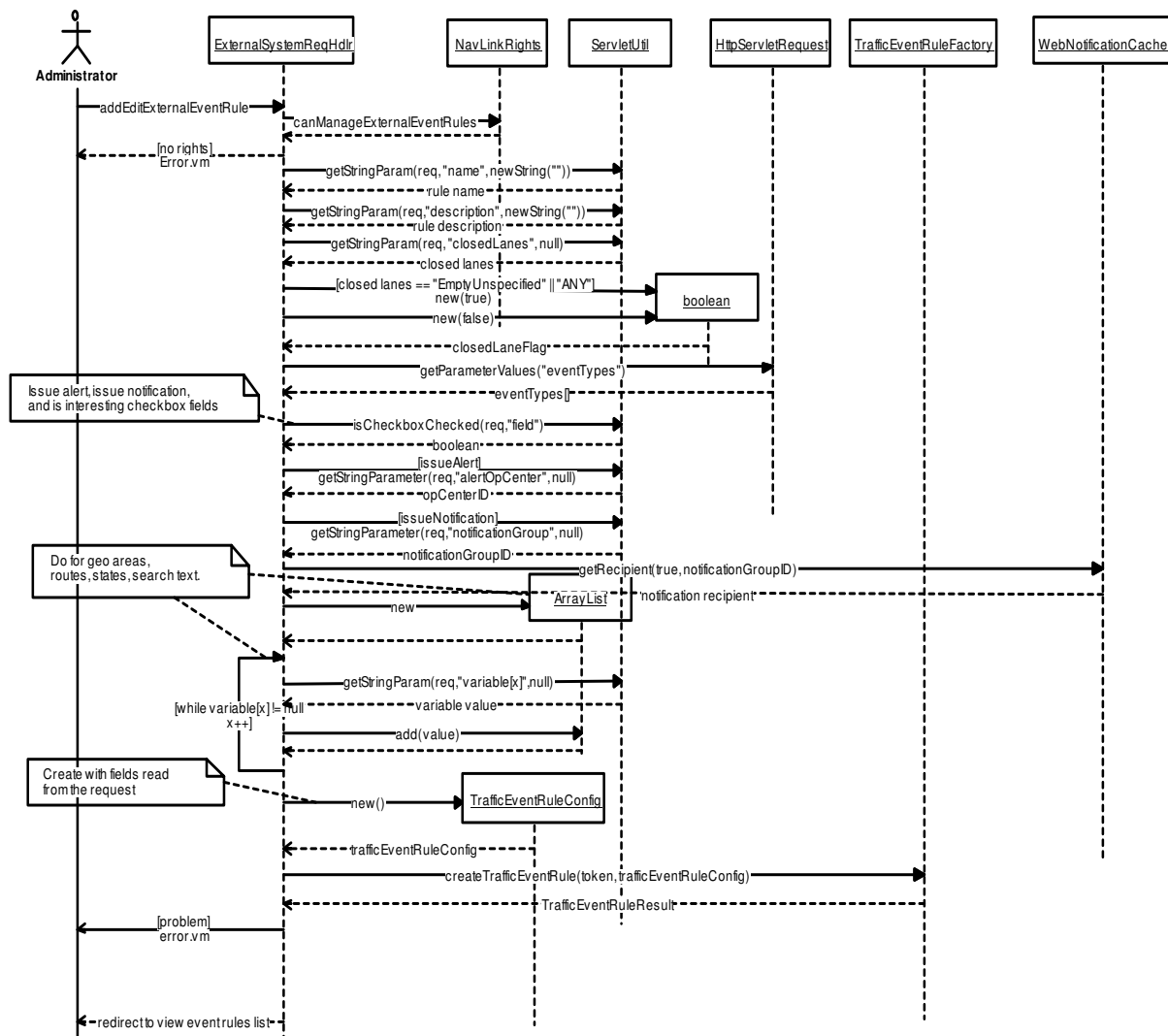


Figure 5-419. ExternalSystemReqHdlr:addEditTrafficEventInclusionRule (Sequence Diagram)

5.51.2.5 ExternalSystemReqHdr:downloadPrivateKey (Sequence Diagram)

This diagram shows the processing that occurs when an administrator chooses to download the private key file for the generated public key. This request sends the encoded bytes of the private key and returns null to continue displaying the client form.

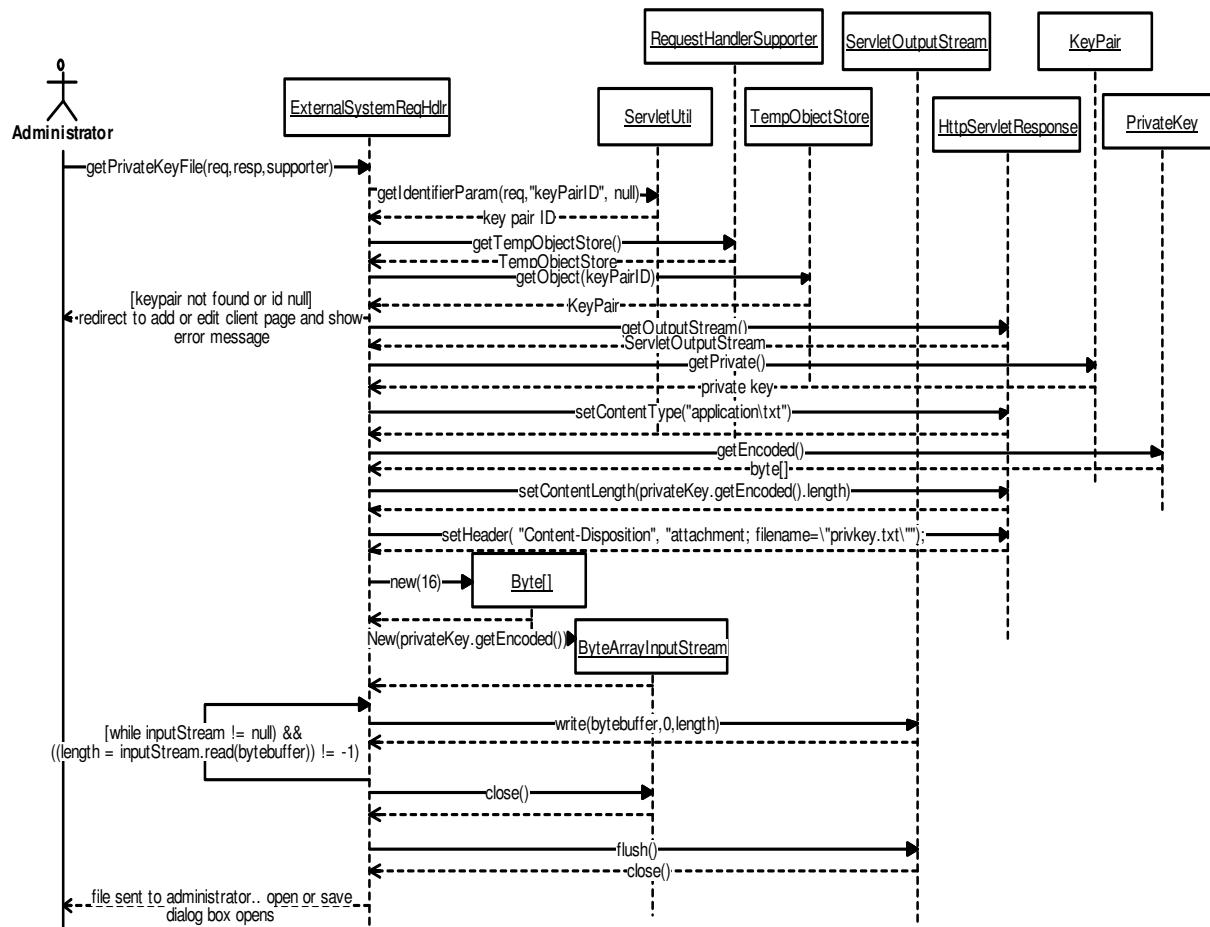


Figure 5-420. ExternalSystemReqHdr:downloadPrivateKey (Sequence Diagram)

5.51.2.6 ExternalSystemReqHdlr:generateKeyPair (Sequence Diagram)

This diagram shows the processing that occurs when generating a public/private keypair. A user manager wrapper is called to generate the KeyPair on the CHART Server. The server returns the generated Key Pair.

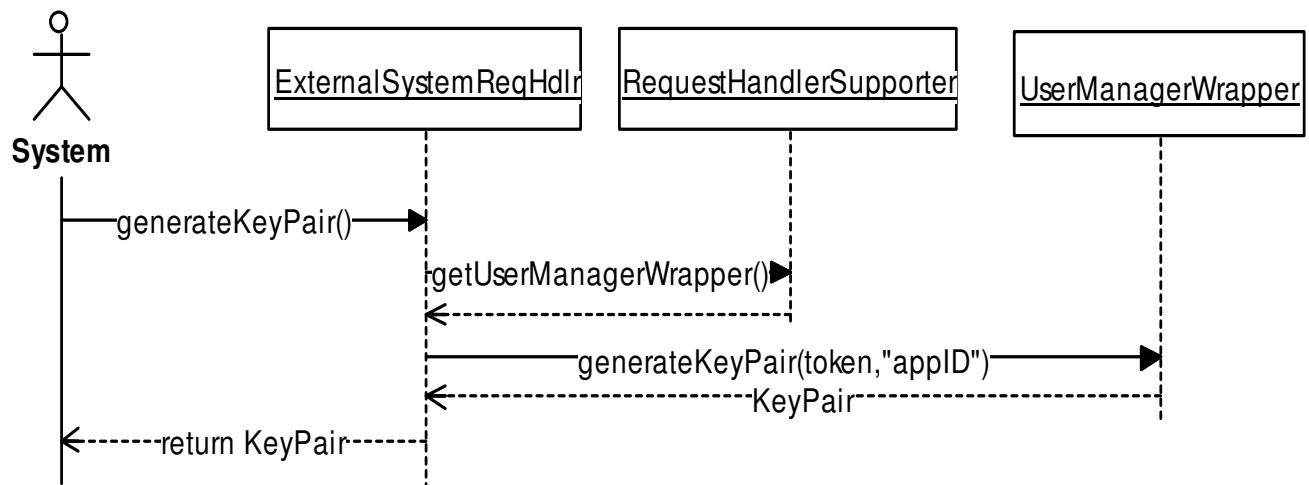


Figure 5-421. ExternalSystemReqHdlr:generateKeyPair (Sequence Diagram)

5.51.2.7 ExternalSystemReqHdlr:getAddEditExternalClientForm (Sequence Diagram)

This diagram shows the processing that occurs when an Administrator adds or updates an external system client.

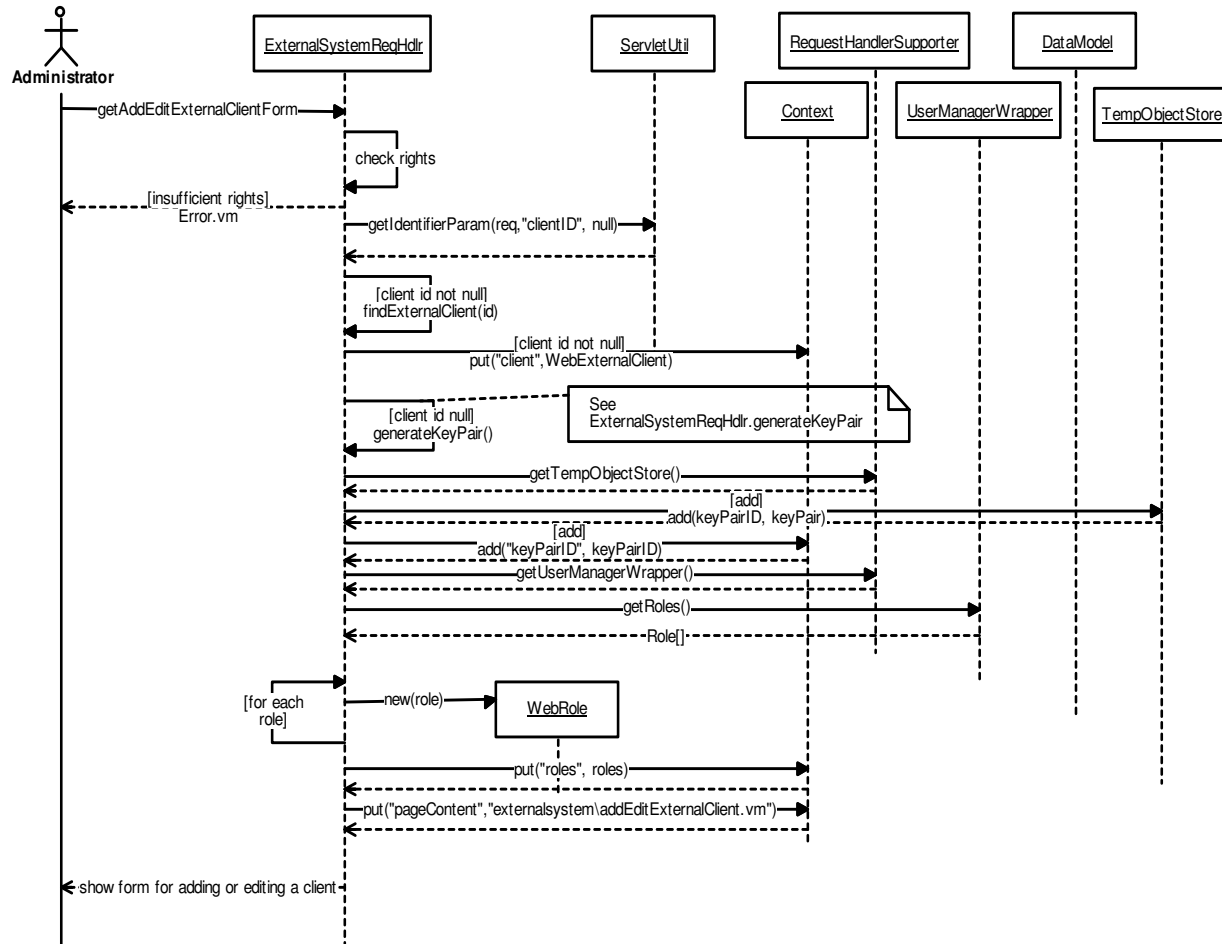


Figure 5-422. ExternalSystemReqHdlr:getAddEditExternalClientForm (Sequence Diagram)

5.51.2.8 ExternalSystemReqHdlr:getAddEditTrafficEventInclusionRuleForm (Sequence Diagram)

This diagram shows the processing that occurs when an administrator has chosen to add a new traffic event inclusion rule, or edit an existing rule.

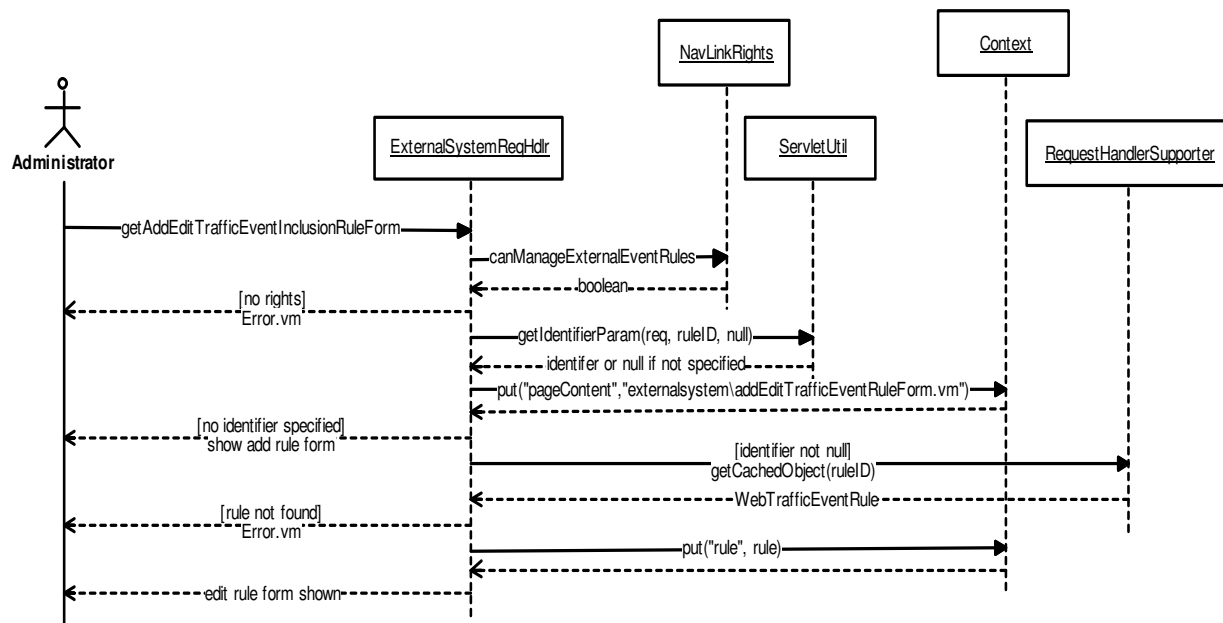


Figure 5-423. ExternalSystemReqHdlr:getAddEditTrafficEventInclusionRuleForm (Sequence Diagram)

5.51.2.9 ExternalSystemReqHdlr:getExternalDeviceQueryForm (Sequence Diagram)

This diagram shows the processing that occurs when an administrator has chosen to submit an external device query to the system. For R3B3, the administrator can query external DMS's and TSS's

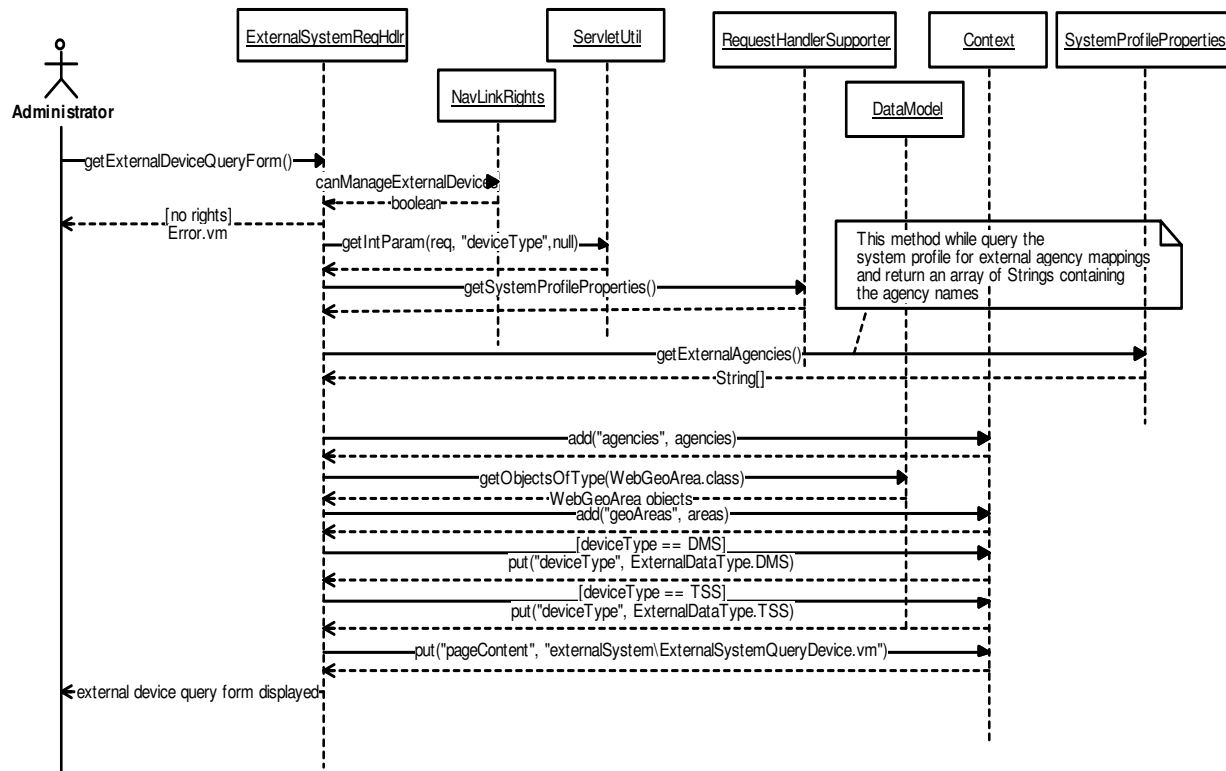


Figure 5-424. ExternalSystemReqHdlr:getExternalDeviceQueryForm (Sequence Diagram)

5.51.2.10 ExternalSystemReqHdlr:removeExternalClient (Sequence Diagram)

This diagram shows the processing that occurs when an external client is removed from the system.

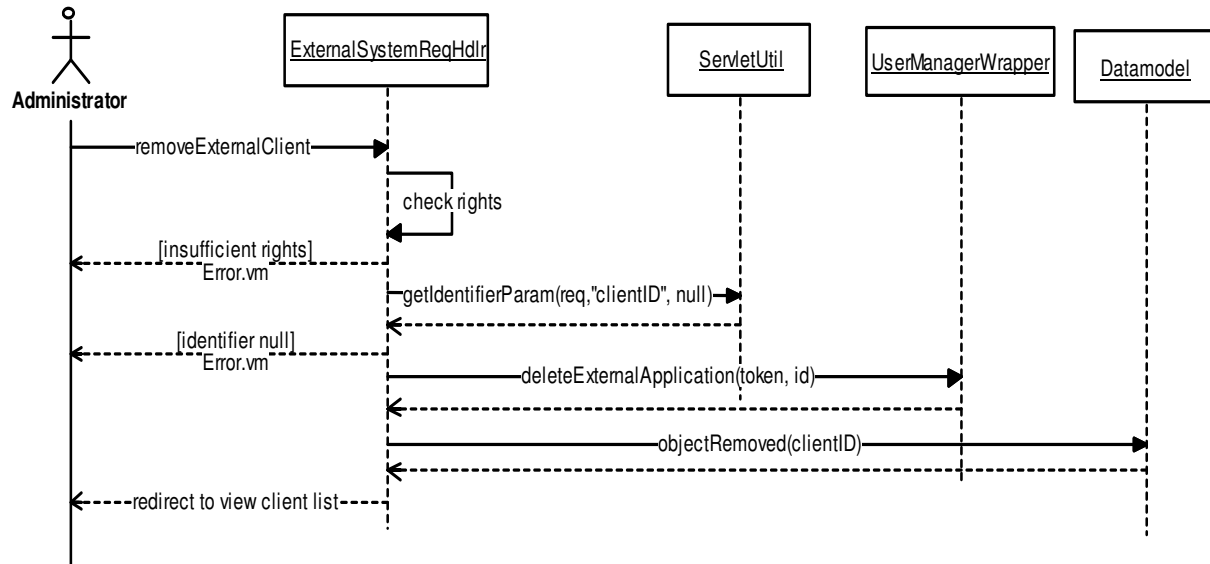


Figure 5-425. ExternalSystemReqHdlr:removeExternalClient (Sequence Diagram)

5.51.2.11 ExternalSystemReqHdlr:removeTrafficEventInclusionRule (Sequence Diagram)

This diagram shows the processing that occurs when an external traffic event inclusion rule is removed.

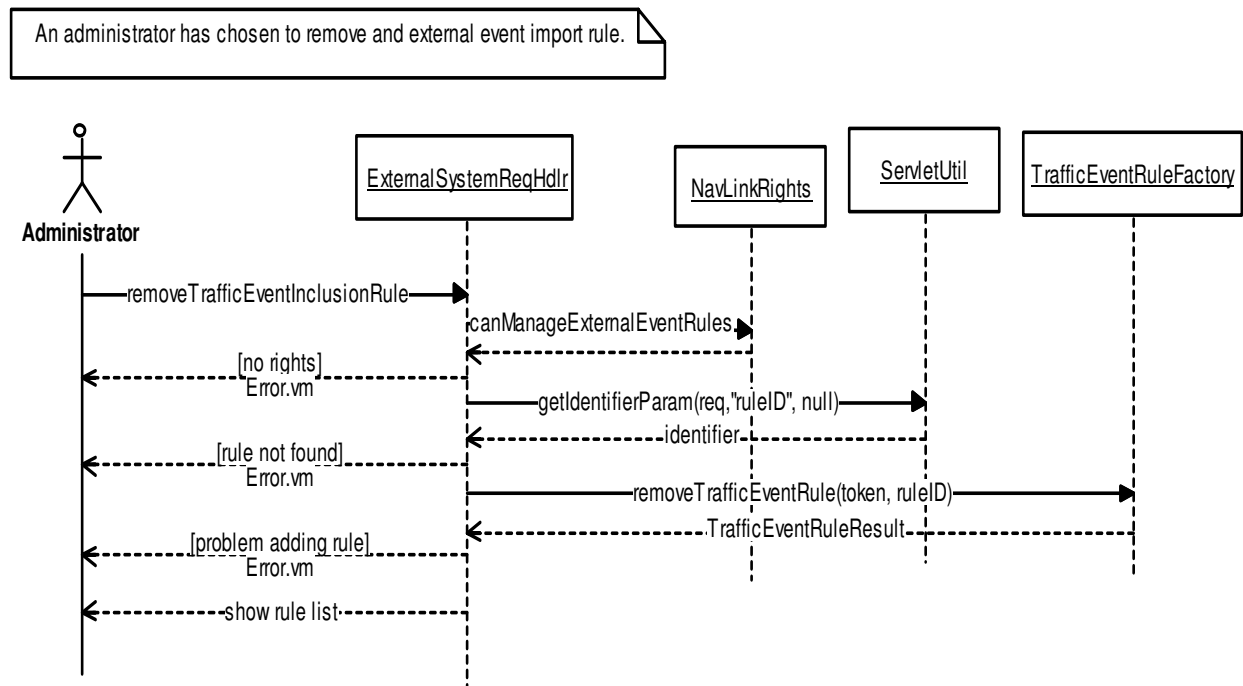


Figure 5-426. ExternalSystemReqHdlr:removeTrafficEventInclusionRule (Sequence Diagram)

5.51.2.12 ExternalSystemReqHdr:submitExternalDeviceQueryForm (Sequence Diagram)

This diagram shows the processing that occurs when an administrator has submitted a query to search for external device candidates. The result is a page containing devices that match the query.

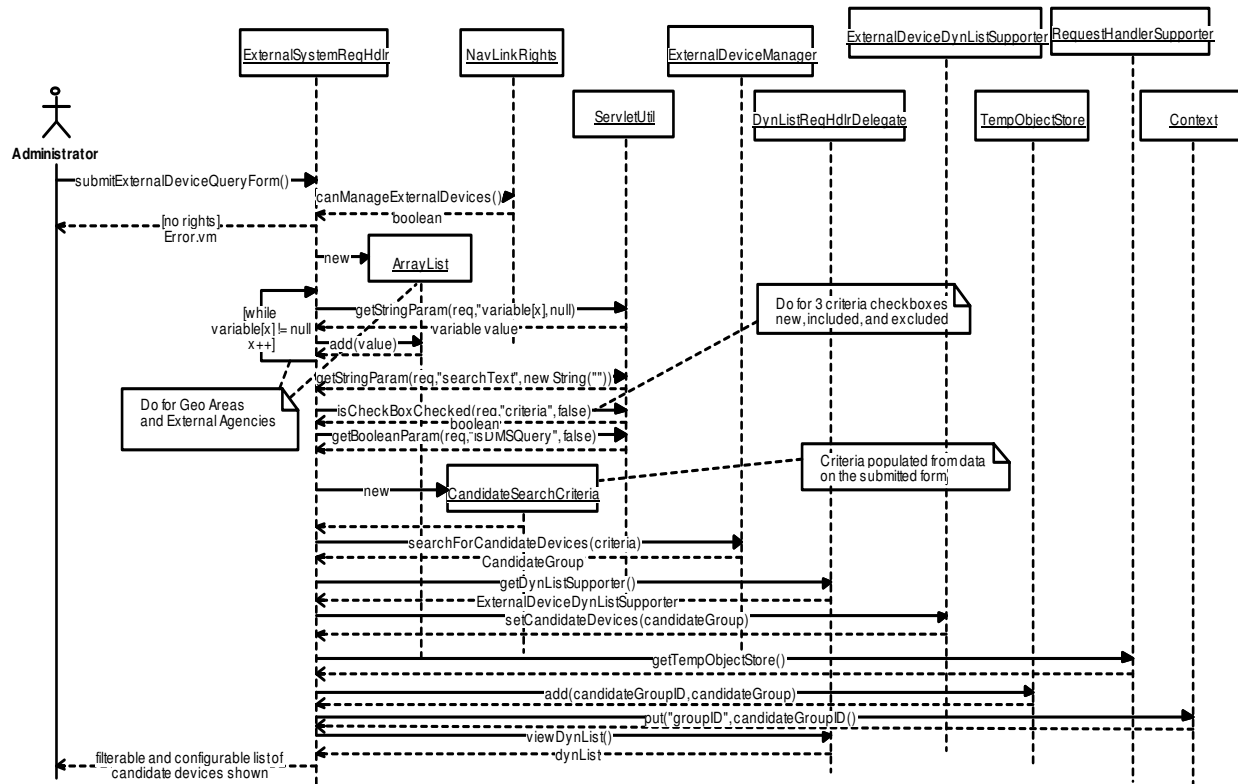


Figure 5-427. ExternalSystemReqHdr:submitExternalDeviceQueryForm (Sequence Diagram)

5.51.2.13 ExternalSystemReqHdr:submitExternalDeviceSelectionForm (Sequence Diagram)

This diagram shows the processing that occurs when an administrator has selected devices to include/exclude from the CHART system.

This diagram shows the processing that occurs when an administrator has selected devices to include/exclude from the CHART system and submitted the form.

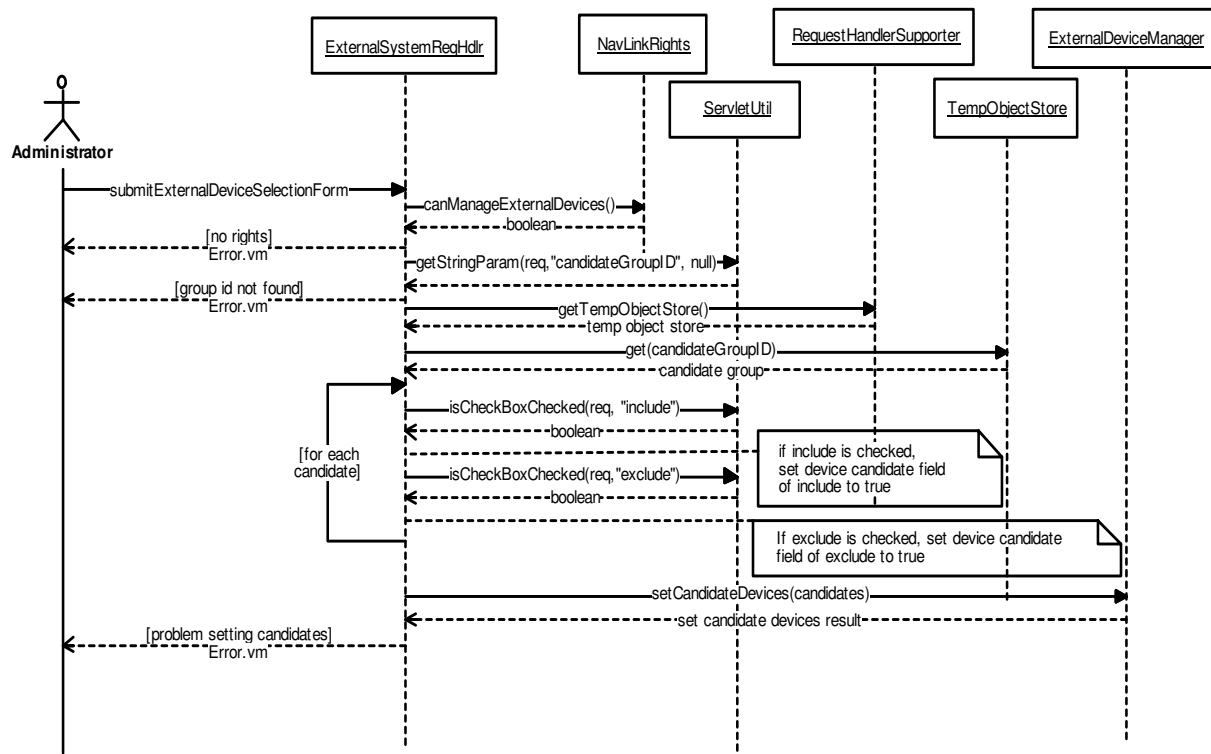


Figure 5-428. ExternalSystemReqHdr:submitExternalDeviceSelectionForm (Sequence Diagram)

5.51.2.14 ExternalSystemReqHdlr:viewExternalClientList (Sequence Diagram)

This diagram shows the processing that occurs when an Administrator views the list of external clients.

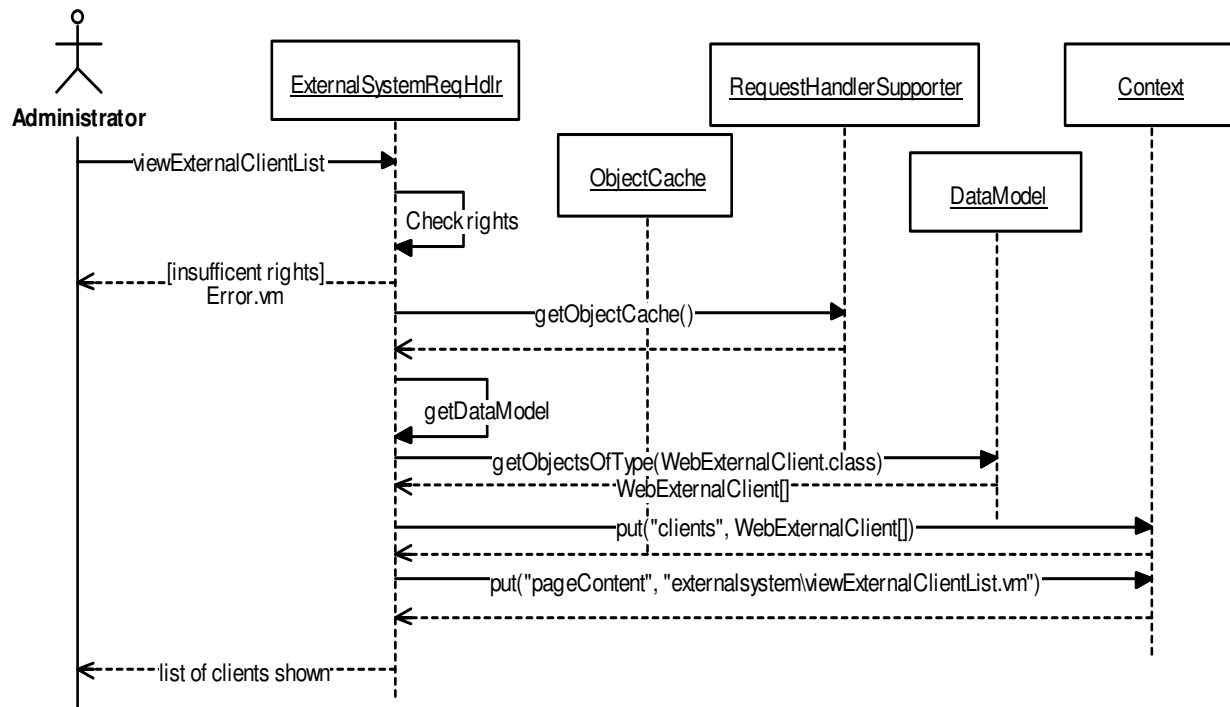


Figure 5-429. ExternalSystemReqHdlr:viewExternalClientList (Sequence Diagram)

5.51.2.15 ExternalSystemReqHdlr:viewExternalSystemConnectionStatus (Sequence Diagram)

This diagram shows the processing that occurs when an operator clicks on the external connections link on the homepage. Connection status is shown for all of the external connections.

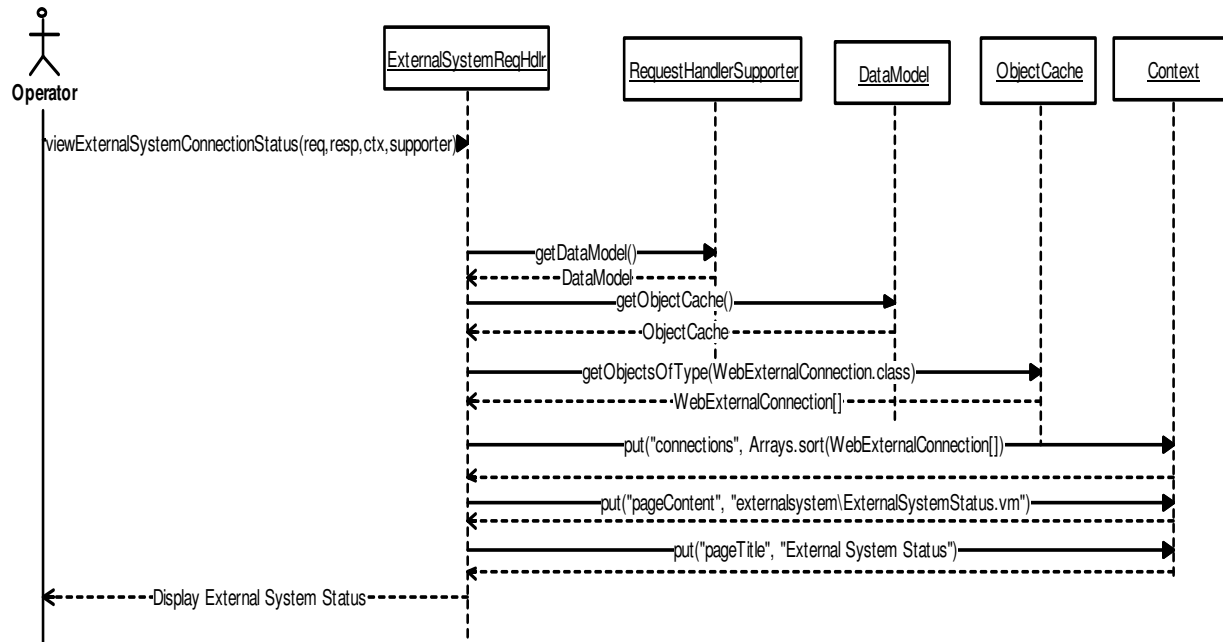


Figure 5-430. ExternalSystemReqHdlr:viewExternalSystemConnectionStatus (Sequence Diagram)

5.51.2.16 ExternalSystemReqHdlr:viewTrafficEventInclusionRules (Sequence Diagram)

This diagram shows the processing that occurs when the administrator views the list of traffic event inclusion rules.

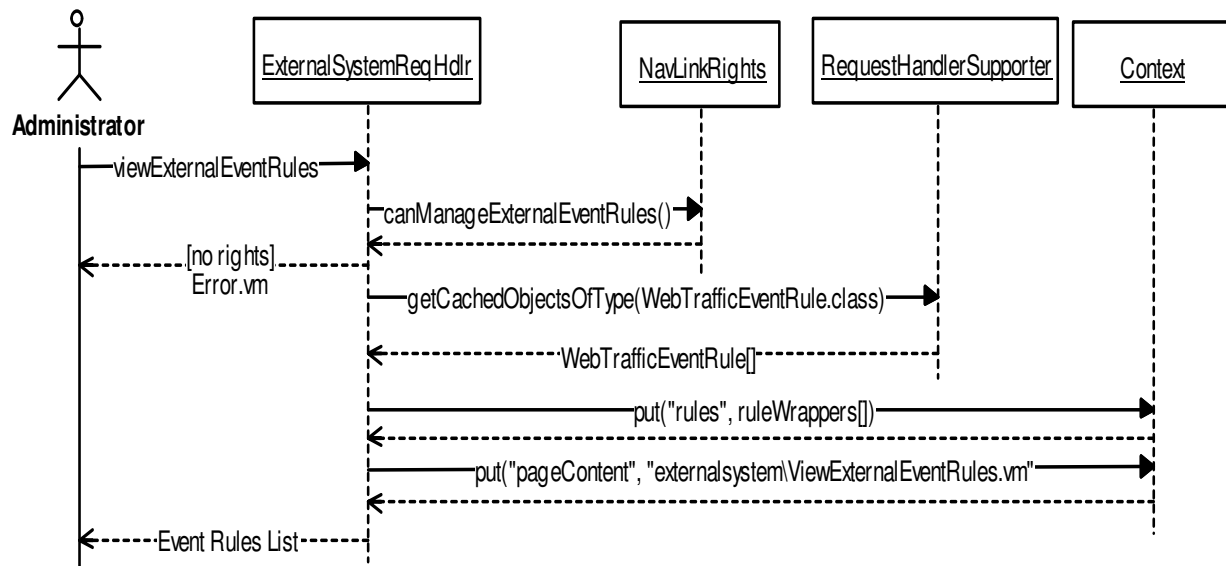


Figure 5-431. ExternalSystemReqHdlr:viewTrafficEventInclusionRules (Sequence Diagram)

5.52 chartlite.servlet.messagesemplates

5.52.1 Class Diagrams

5.52.1.1 DMSTravInfoMsgTemplateDynListClasses (Class Diagram)

This diagram contains classes related to the display of message templates in a dynamic (sortable/filterable) list.

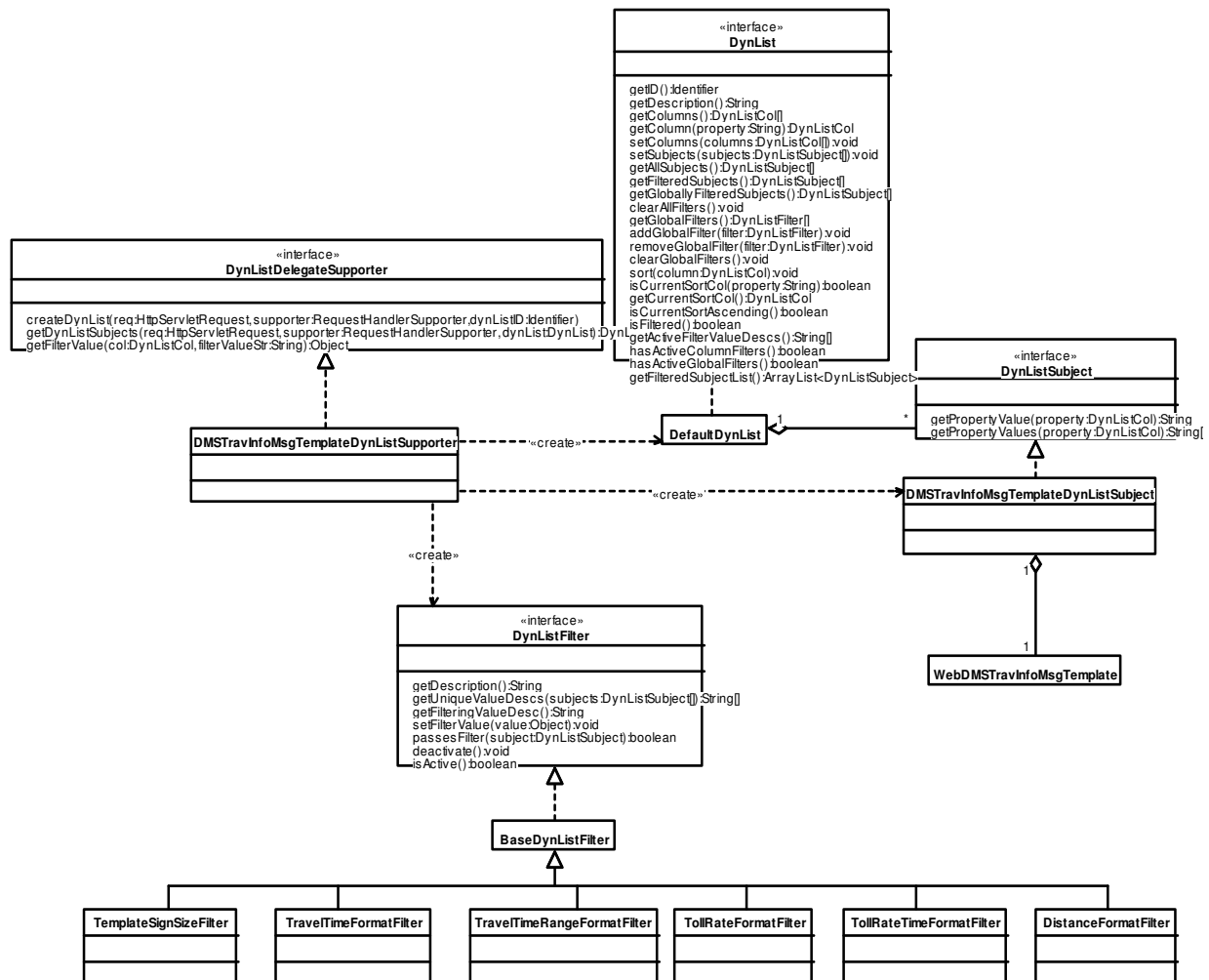


Figure 5-432. DMSTravInfoMsgTemplateDynListClasses (Class Diagram)

5.52.1.1.1 BaseDynListFilter (Class)

This abstract class provides a base implementation of the DynListFilter interface.

5.52.1.1.2 DefaultDynList (Class)

This class provides a default implementation of the DynList interface. It supports a collection of columns, a collection of global filters, and a collection of subjects. Filters in this list are treated additively - that is, a subject must pass all filters to be displayed.

5.52.1.1.3 DistanceFormatFilter (Class)

This class allows filtering on the distance format attribute of a DMS message template.

5.52.1.1.4 DMSTravInfoMsgTemplateDynListSubject (Class)

This class is a proxy representing a single DMS message template in a dynamic list. It has a reference to the template object.

5.52.1.1.5 DMSTravInfoMsgTemplateDynListSupporter (Class)

This class provides supporting functionality for displaying a DMS message template dynamic list. It creates the list object and gets the objects representing templates to display in the list.

5.52.1.1.6 DynList (Class)

This interface is implemented by classes that wish to provide dynamic list capabilities. A dynamic list is a list of items that has one or more columns that can optionally be sorted, and the list can be filtered by column values or by global filters.

5.52.1.1.7 DynListDelegateSupporter (Class)

This interface contains functionality to support the DynListReqHdlrDelegate

5.52.1.1.8 DynListFilter (Class)

This interface is implemented by classes that are used to filter dynamic lists.

5.52.1.1.9 DynListSubject (Class)

This interface is implemented by classes that wish to be capable of being displayed in a dynamic list.

5.52.1.1.10TemplateSignSizeFilter (Class)

This class allows filtering on the sign size attribute of a DMS message template.

5.52.1.1.11TollRateFormatFilter (Class)

This class allows filtering on the toll rate format attribute of a DMS message template.

5.52.1.1.12TollRateTimeFormatFilter (Class)

This class allows filtering on the toll rate time format attribute of a DMS message template.

5.52.1.1.13TravelTimeFormatFilter (Class)

This class allows filtering on the travel time format attribute of a DMS message template.

5.52.1.1.14TravelTimeRangeFormatFilter (Class)

This class allows filtering on the travel time range format attribute of a DMS message template.

5.52.1.1.15WebDMSTravInfoMsgTemplate (Class)

This class wraps the DMSTravInfoMsgTemplate CORBA object representing a message template. It caches the data represented by the remote object and provides accessors for easy access to the cached data.

5.52.1.2 GUIMessageTemplateServletClasses (Class Diagram)

This diagram shows GUI classes that are involved in handling requests related to traveler information message templates.

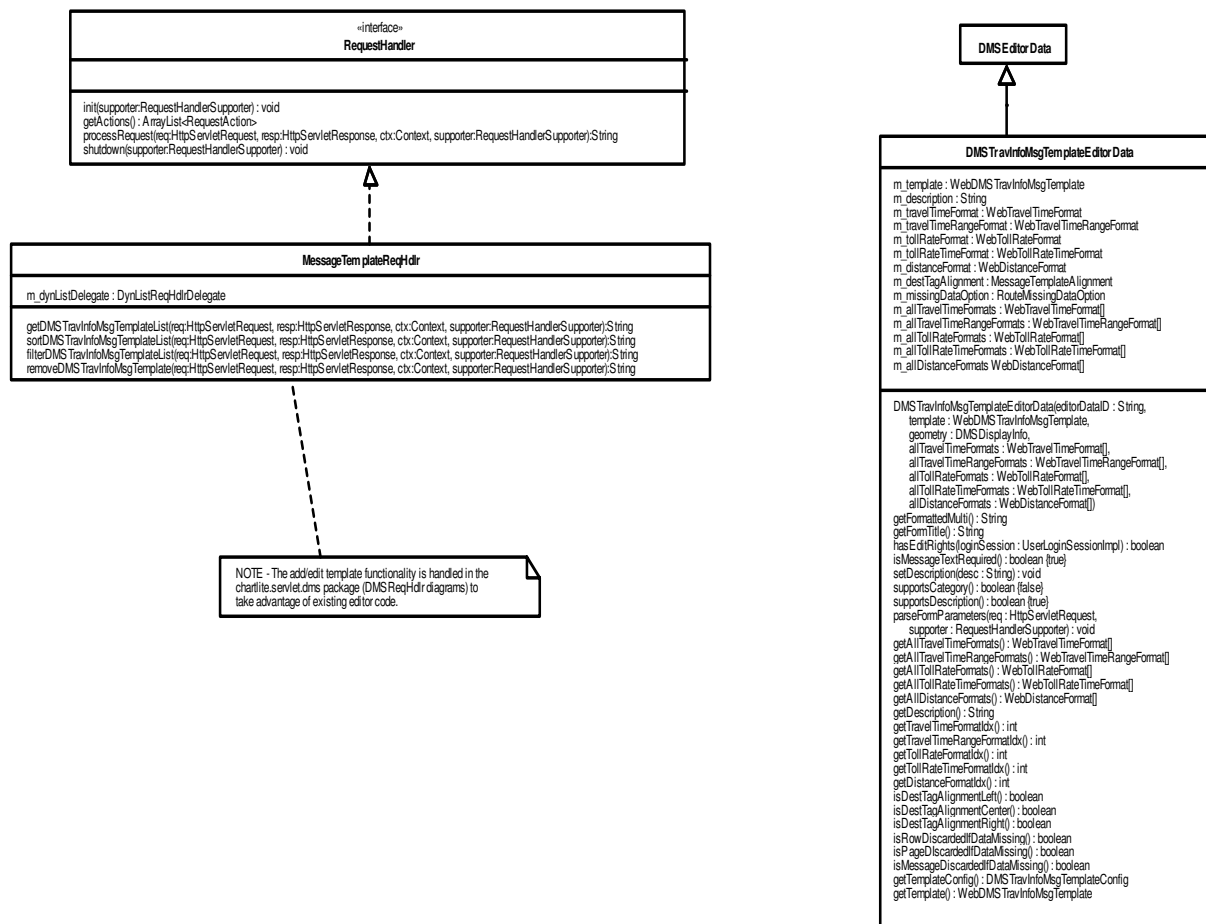


Figure 5-433. GUIMessageTemplateServletClasses (Class Diagram)

5.52.1.2.1 DMSEditorData (Class)

This class represents an instance of a DMS message being edited in an editor. It provides storage so that the message and editor state can be preserved during interim requests before the form is submitted. It also has logic for manipulating the editor session. This is a base class and will be extended for specific editor types.

5.52.1.2.2 DMSTravInfoMsgTemplateEditorData (Class)

This class contains the data related to one instance of the DMS message template editor. It extends DMSEditorData to provide common editor functionality, and also has methods for getting and setting the editor data.

5.52.1.2.3 MessageTemplateReqHdlr (Class)

This class handles requests related to message templates, except for the requests for adding / editing a template, which are in the DMSReqHdlr class.

5.52.1.2.4 RequestHandler (Class)

This interface specifies methods that are to be implemented by classes that are used to process requests.

5.52.2 Sequence Diagrams

5.52.2.1 MessageTemplateReqHdr:filterDMSTravInfoMsgTemplateList (Sequence Diagram)

This diagram shows the processing to filter the list of DMS message templates. The work is mostly delegated to DynListReqHdrDelegate, which uses the list ID to retrieve the existing list from the temporary object store. If not found (i.e., if the list has timed out) it will create a new list and add it to the temporary object store. The filter property is used to retrieve the DynListCol object representing the column to filter on, and its DynListFilter is retrieved. The DynListSupporter is called to get the filter value for the column, based on the filter value string from the request. If the filter value is "--Any--" or there is no filter value object returned, the column filter is deactivated; otherwise, the filter value is set into the filter for later use. A redirect is sent in the response so the browser will issue another request to view the list. When building the new list to display, the Velocity code will ask for the filtered list and the filter will be applied at that time.

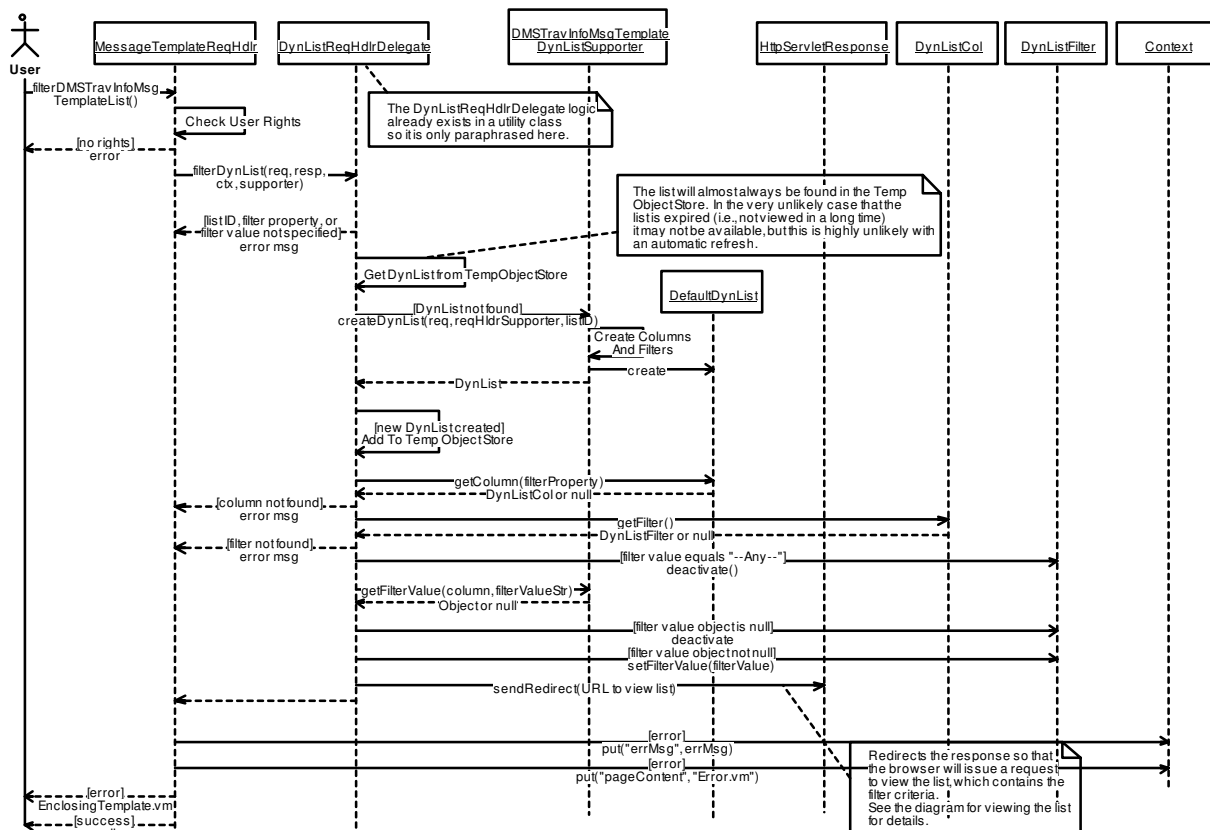


Figure 5-434. MessageTemplateReqHdr:filterDMSTravInfoMsgTemplateList (Sequence Diagram)

5.52.2.2 MessageTemplateReqHdr:getDMSTravInfoMsgTemplateList (Sequence Diagram)

This diagram shows the processing to show the list of DMS message templates. The work is mostly delegated to DynListReqHdrDelegate, which uses the list ID (if specified) to retrieve the existing list from the temporary object cache. If not found (i.e., if the list is displayed for the first time or the object has timed out) it will call the DMSTravInfoMsgTemplateDynListSupporter class to create a new list. The list is created and added to the temporary object cache, and the response is sent to redirect the browser to view the list again. If the list already existed, the DMSTravInfoMsgTemplate objects are retrieved from the factory wrapper and new DMSTravInfoMsgTemplateDynListSubject objects are created to represent them in the list. These subjects are stored in the dynamic list for later use. The previous filters are cleared (if requested) and the Velocity context is prepared so that the Velocity template can render the HTML for the page.

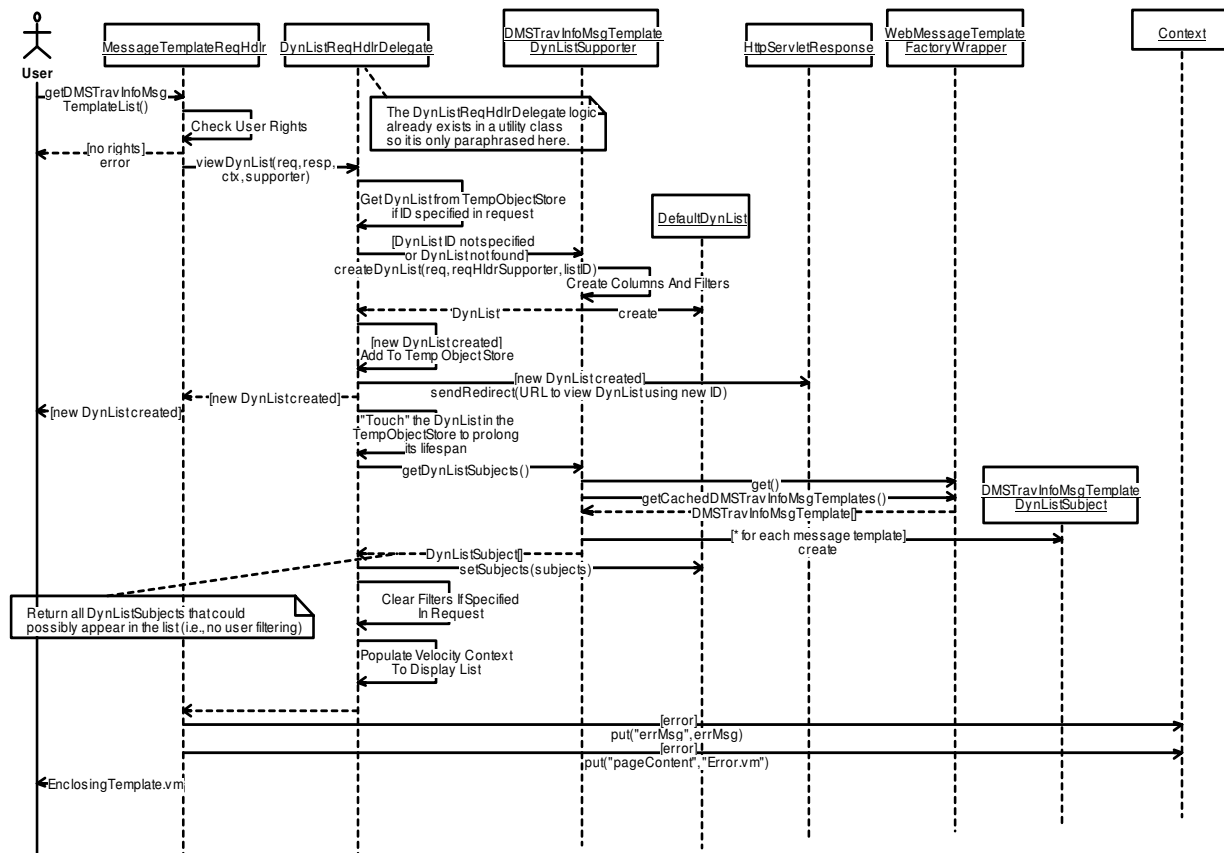


Figure 5-435. MessageTemplateReqHdr:getDMSTravInfoMsgTemplateList (Sequence Diagram)

5.52.2.3 MessageTemplateReqHdlr:removeDMSTravInfoMsgTemplate (Sequence Diagram)

This diagram shows the processing to remove a DMS message template from the system. The templateID parameter is used to get the WebDMSTravInfoMsgTemplate object from the factory wrapper cache. The WebDMS objects are retrieved from the cache and each one is asked if it is using the template with the given template ID. If some DMSs are using the template, an error message is shown to the user. Otherwise, the DMSTravInfoMsgTemplate CORBA object is called to remove itself. If successful (or an OBJECT_NOT_EXIST error occurs) the template is removed from the factory wrapper cache, and the response is sent to redirect the browser to show the updated list of templates.

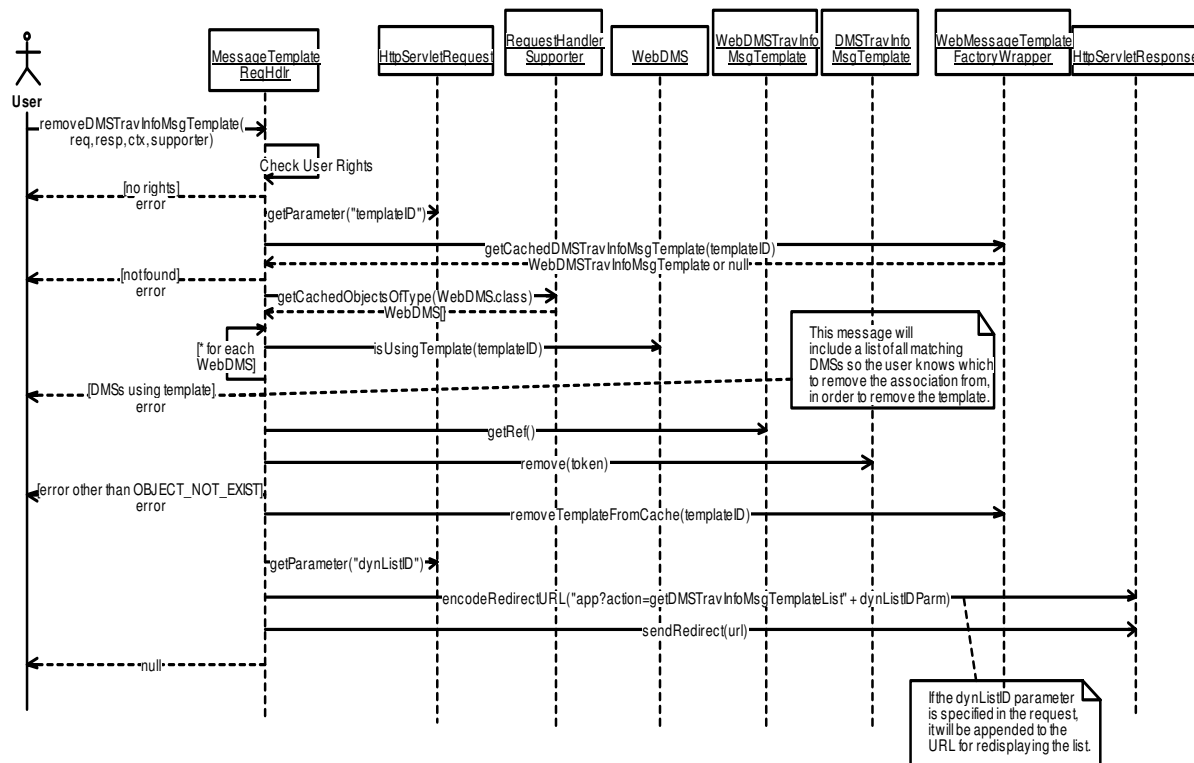


Figure 5-436. MessageTemplateReqHdlr:removeDMSTravInfoMsgTemplate (Sequence Diagram)

5.52.2.4 MessageTemplateReqHdlr:sortDMSTravInfoMsgTemplateList (Sequence Diagram)

This diagram shows the processing to sort the list of DMS message templates. The work is mostly delegated to DynListReqHdlrDelegate, which uses the list ID to retrieve the existing list from the temporary object store. If not found (i.e., if the list has timed out) it will create a new list and add it to the temporary object store. The sort property is used to retrieve the DynListCol object representing the column to sort on, and the dynamic list is called to sort the objects on that column, rearranging the objects stored in the list object. A redirect is sent in the response so the browser will issue another request to view the (newly sorted) list.

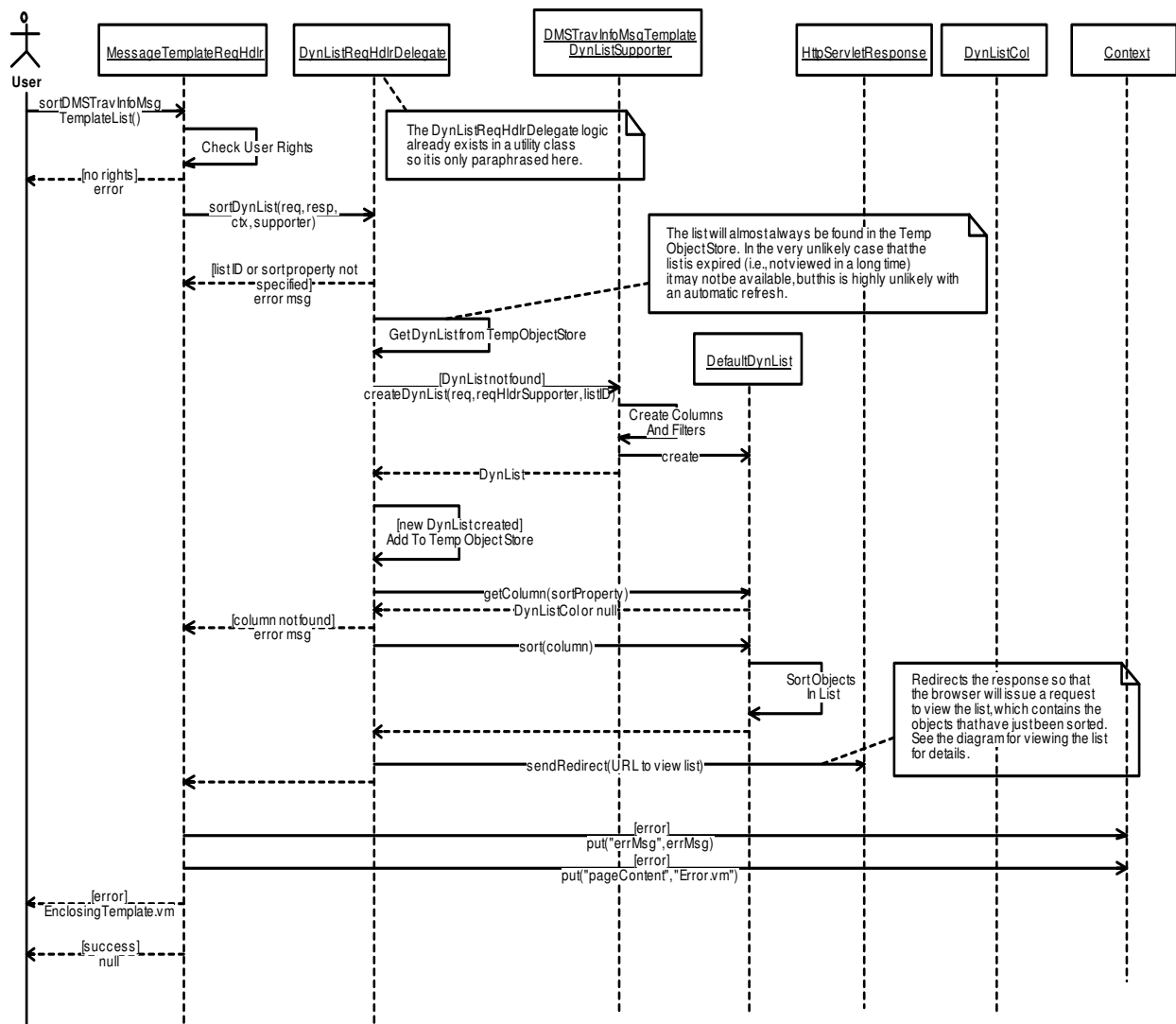


Figure 5-437. MessageTemplateReqHdlr:sortDMSTravInfoMsgTemplateList (Sequence Diagram)

5.53 Chartlite.servlet.video

5.53.1 Class Diagrams

5.53.1.1 GUIVideoServletClasses (Class Diagram)

This diagram shows GUI classes involved in processing video-related requests.

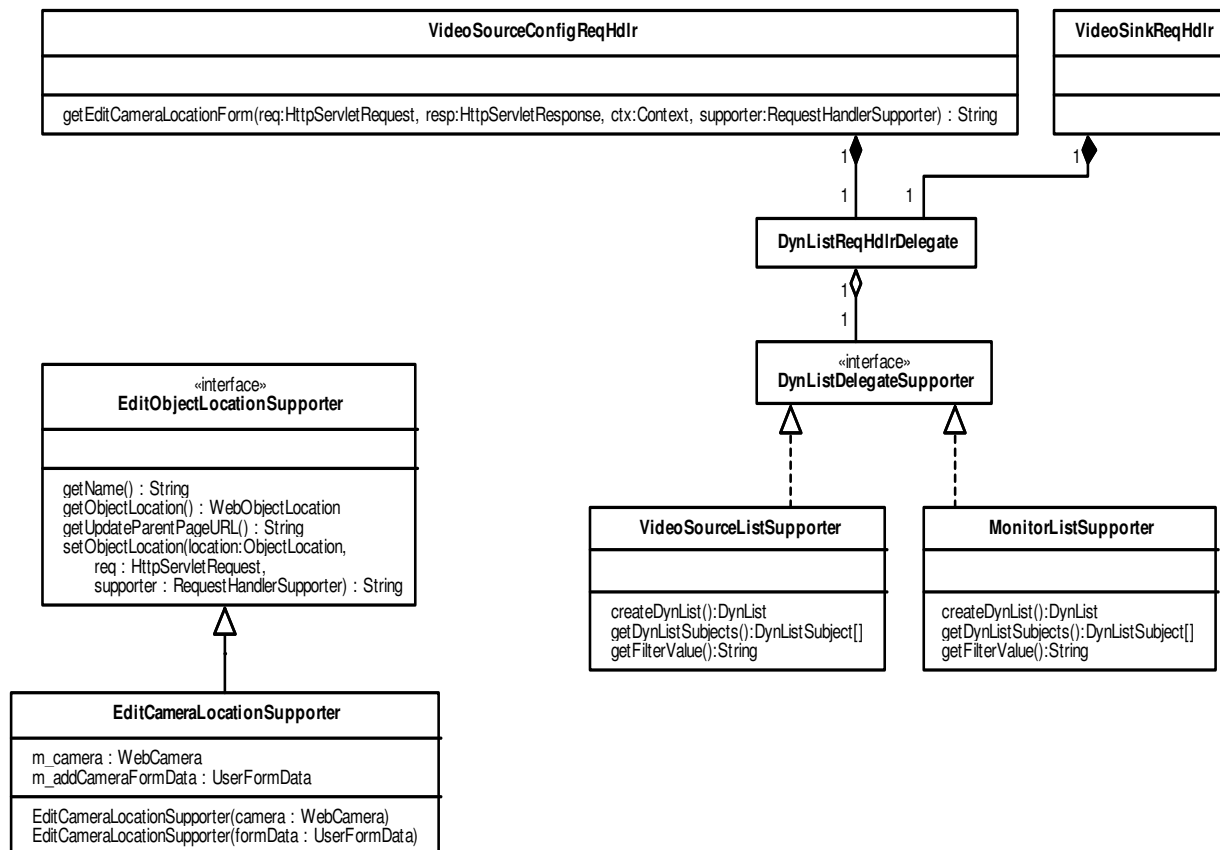


Figure 5-438. GUIVideoServletClasses (Class Diagram)

5.53.1.1.1 DynListDelegateSupporter (Class)

This interface contains functionality to support the DynListReqHdlrDelegate

5.53.1.1.2 DynListReqHdlrDelegate (Class)

This class helps request handlers support dynamic lists. Requests to view, sort, or filter dynamic lists can be passed from a request handler to this class, provided the URL used for the requests contain parameters required by this class, such as the id of the list, the property

name, and/or the filter value.

5.53.1.1.3 EditCameraLocationSupporter (Class)

This class is used to support editing the location of an existing or new VideoCamera.

5.53.1.1.4 EditObjectLocationSupporter (Class)

This interface provides functionality allowing the location data to be edited. (For example, the target of the edited location may be an existing object, or it may be a form data object for creating a new object).

5.53.1.1.5 MonitorListSupporter (Class)

This class is a DynListDelegateSupporter that provides Monitor specific functionality to the generic DynListReqHdlrDelegate.

5.53.1.1.6 VideoSinkReqHdlr (Class)

This class is a request handler that processes requests related to video sinks such as Monitors.

5.53.1.1.7 VideoSourceConfigReqHdlr (Class)

This class handles requests related to video source configuration.

5.53.1.1.8 VideoSourceListSupporter (Class)

This class is a DynListDelegateSupporter that provides Video Source specific functionality to the generic DynListReqHdlrDelegate.

5.53.2 Sequence Diagrams

5.53.2.1 EditCameraLocationSupporter:setObjectLocation (Sequence Diagram)

This diagram shows the processing to save the camera location when the user submits the Edit Location form. The SpecifyLocationReqHdlr calls the EditCameraLocationSupporter with the location parsed from the request. If the location is being edited while adding / copying a camera, a utility method is called to store the location in the UserFormData that has already been stored in the TempObjectStore. When the Add/Copy form is redisplayed, the UserFormData will be used to populate the location hidden form parameters in the Add/Copy form. If the location is being saved for an existing DMS, the WebCamera is called to get the VideoCamera reference, the configuration is queried from the camera, the location is modified within the configuration, and the VideoCamera is called to set the configuration. Since this is an asynchronous command, the URL to view the command status is saved in the EditCameraLocationSupporter and will be passed back in the XML response so that the parent window's URL can be set.

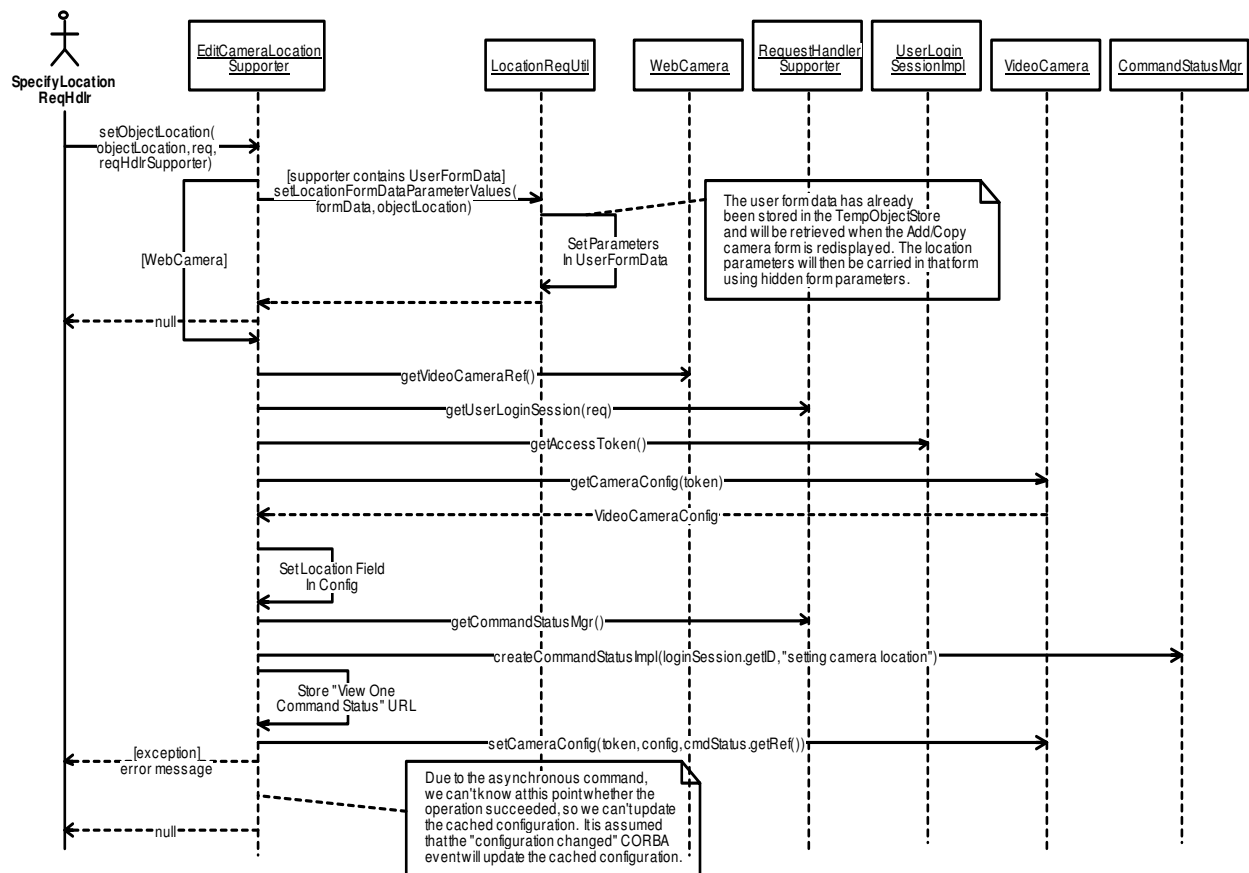


Figure 5-439. EditCameraLocationSupporter:setObjectLocation (Sequence Diagram)

5.53.2.2 MonitorListSupporter:createDynList (Sequence Diagram)

This diagram shows the processing that is performed when the MonitorListSupporter is called to create a Monitor dynamic list. A DefaultDynListCol object is created for each column and stored in an ArrayList. A BaseDynListFilter derived object is added to the column if the column supports filtering. The list of columns is used to construct a DefaultDynList. If the "remote" parameter is present in the request, a RemoteMonitorsDynListFilter is constructed and added to the dyn list as a global filter.

This processing is updated in R3B3 to add support for the Connection Site column.

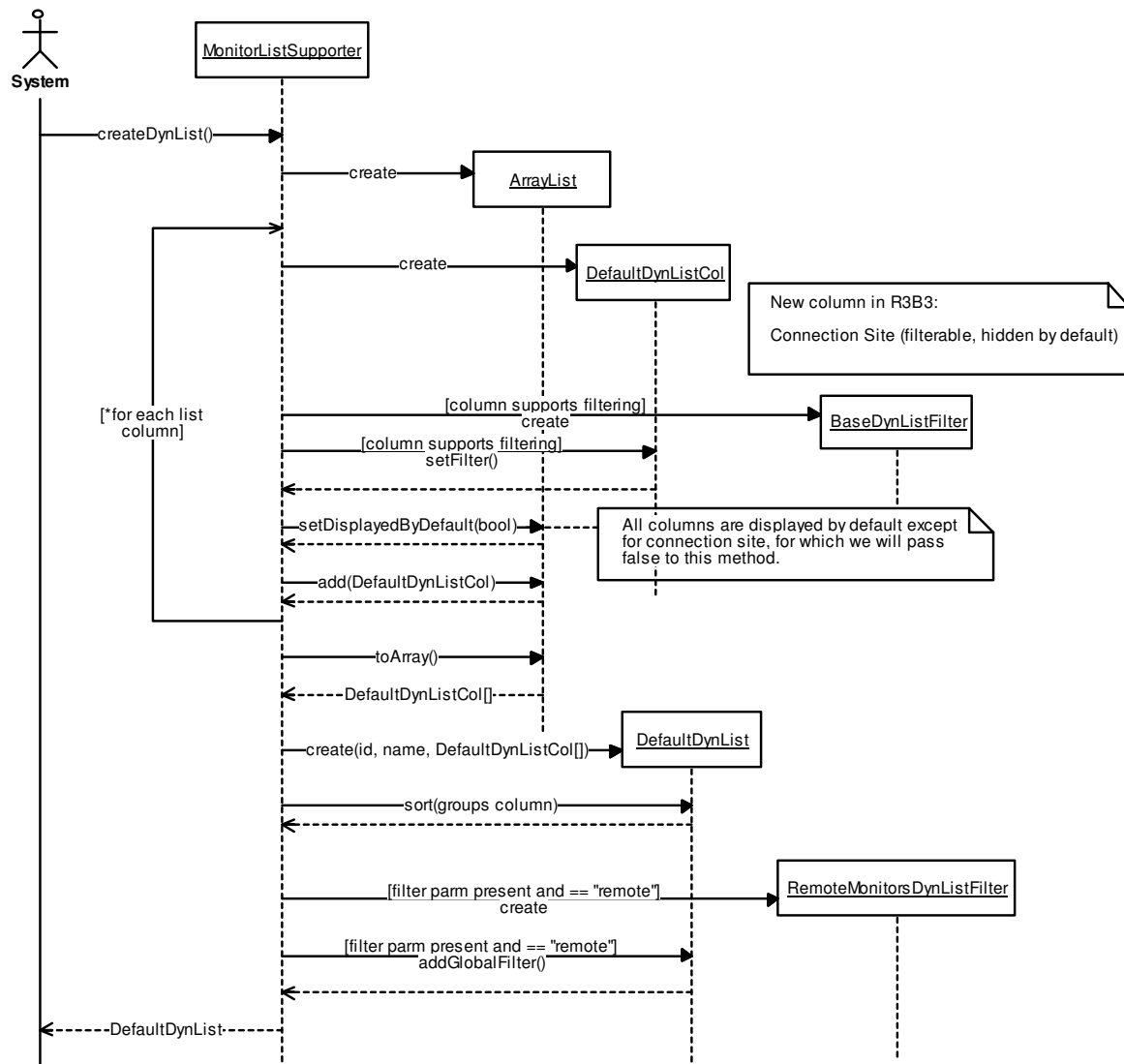


Figure 5-440. MonitorListSupporter:createDynList (Sequence Diagram)

5.53.2.3 VideoSourceConfigReqHdlr:getEditCameraLocationForm (Sequence Diagram)

This diagram shows how the Edit Camera Location form is displayed. The sourceID parameter is parsed from the request. If the parameter is not specified, it is an add/copy operation, but if it is specified, the user is editing the location of an existing camera. If adding/copying a camera, the request parameters from the add/edit form are saved into a UserFormData object that is stored in the TempObjectStore and also in the EditCameraLocationSupporter object that is created. If the source ID is specified, the WebCamera is retrieved from the cache, the user's rights are checked for the given camera's organization, and the EditCameraLocationSupporter object is created. This object is added to the TempObjectStore and the response is redirected so that the displayEditObjectLocationDataForm request is invoked.

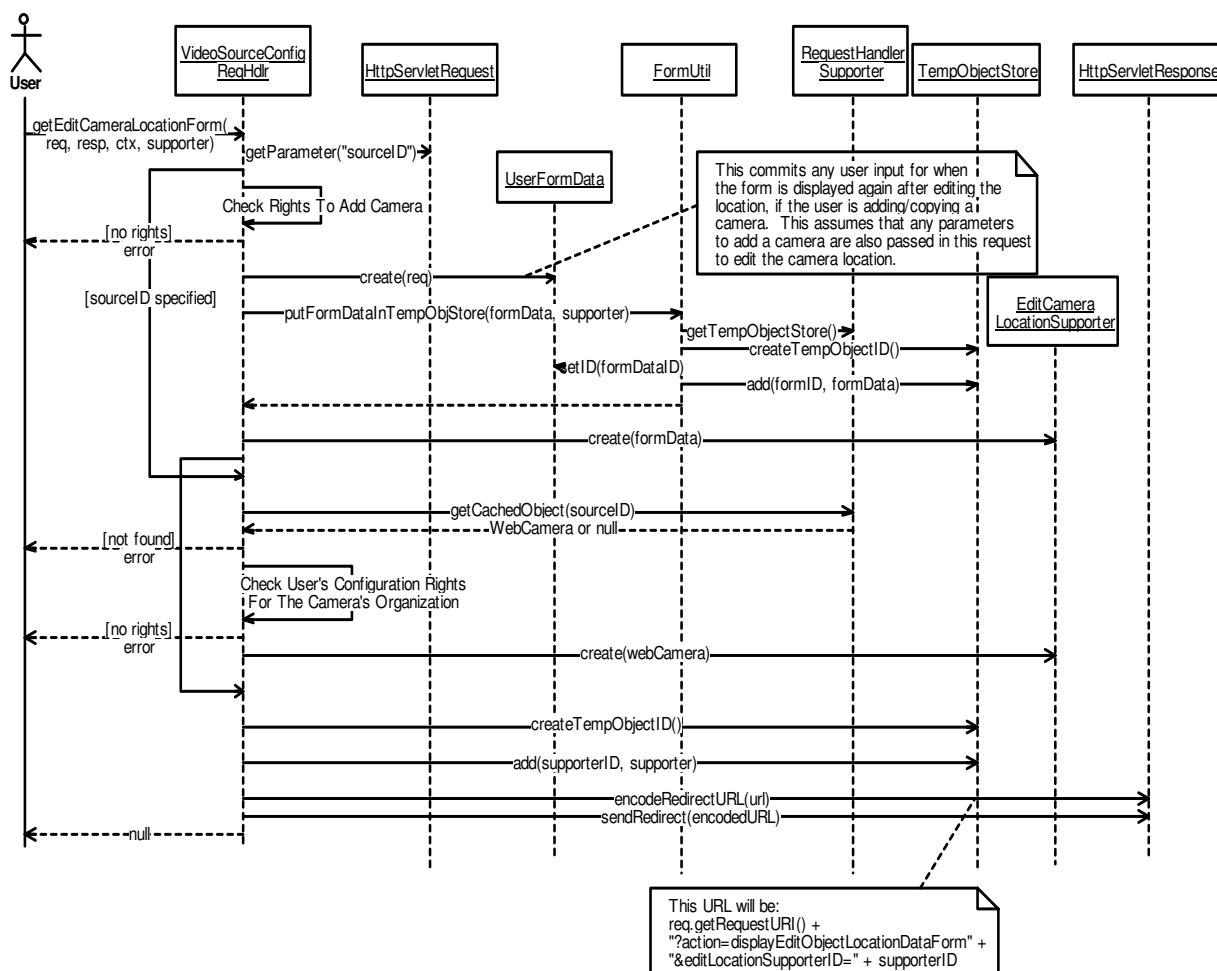


Figure 5-441. VideoSourceConfigReqHdlr:getEditCameraLocationForm (Sequence Diagram)

5.53.2.4 VideoSourceListSupporter:createDynList (Sequence Diagram)

This diagram shows the processing that is performed when the VideoSourceListSupporter is called to create a Video Source dynamic list. A DefaultDynListCol object is created for each column and stored in an ArrayList. A derivation of DynListComparator is constructed and added to each column that requires a custom comparator, and a BaseDynListFilter derived object is added to the column if the column supports filtering. The list of columns is used to construct a VideoSourceList (a subclass of DefaultDynList). The filterType parameter (if present) is used to make other filter related configurations to the dyn list. If the filterType parameter is set to "OpCenterFolders", an OpCenterFolderFilter is constructed, configured for the user's op center, and added to the dyn list as a global filter which will show only Video Sources that exist in folders that are tagged for use with the user's op center. The "region" filterType parameter (if present) is used to set the TextValueColFilter used for filtering the region column to the region specified. Other filterType values are used to set the status filter.

This processing is updated in R3B3 to add support for the following columns: Route, Direction, County, Connection Site, Mile Post.

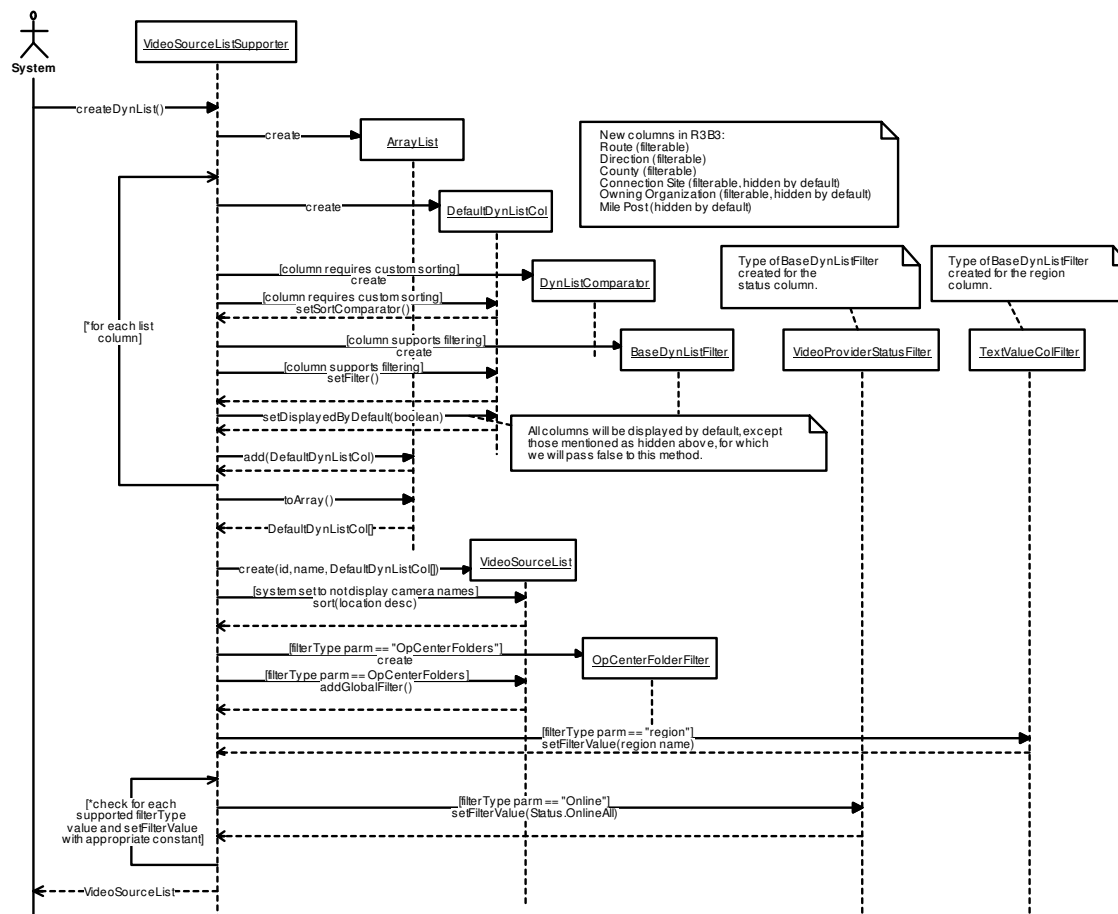


Figure 5-442. VideoSourceListSupporter:createDynList (Sequence Diagram)

5.54 Chartlite.servlet.trafficevents

5.54.1 Class Diagrams

5.54.1.1 GUITrafficEventsDynListClasses (Class Diagram)

This diagram contains classes pertaining to the traffic event dynamic list.

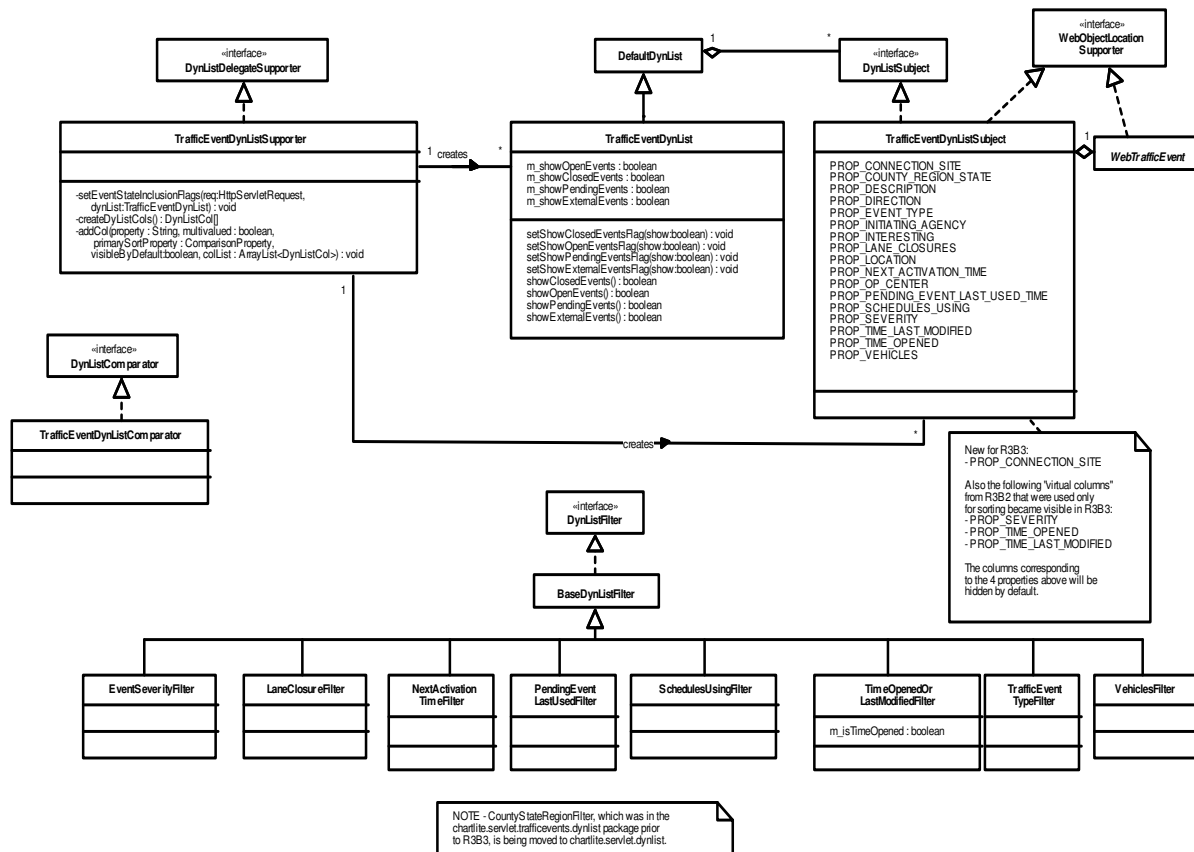


Figure 5-443. GUITrafficEventsDynListClasses (Class Diagram)

5.54.1.1.1 BaseDynListFilter (Class)

This abstract class provides a base implementation of the DynListFilter interface.

5.54.1.1.2 DefaultDynList (Class)

This class provides a default implementation of the DynList interface. It supports a collection of columns, a collection of global filters, and a collection of subjects. Filters in this list are treated additively - that is, a subject must pass all filters to be displayed.

5.54.1.1.3 DynListComparator (Class)

This interface is implemented by classes that are used to sort dynamic lists.

5.54.1.1.4 DynListDelegateSupporter (Class)

This interface contains functionality to support the DynListReqHdlrDelegate

5.54.1.1.5 DynListFilter (Class)

This interface is implemented by classes that are used to filter dynamic lists.

5.54.1.1.6 DynListSubject (Class)

This interface is implemented by classes that wish to be capable of being displayed in a dynamic list.

5.54.1.1.7 EventSeverityFilter (Class)

This filter allows filtering by the severity of the traffic event (i.e., percentage of lanes closed).

5.54.1.1.8 LaneClosureFilter (Class)

This filter allows filtering the traffic event list by the number of lanes closed.

5.54.1.1.9 NextActivation TimeFilter (Class)

This class allows filtering the pending event list by next activation time of the pending event.

5.54.1.1.10 PendingEvent LastUsedFilter (Class)

This class allows filtering the pending event list by last used time of the pending event.

5.54.1.1.11 SchedulesUsingFilter (Class)

This class allows filtering the pending event list by whether the pending event is used by schedules.

5.54.1.1.12 TimeOpenedOr LastModifiedFilter (Class)

This class allows the event list to be filtered by time opened or time last modified.

5.54.1.1.13 TrafficEvent TypeFilter (Class)

This class allows the traffic event list to be filtered by the traffic event type.

5.54.1.1.14 TrafficEventDynList (Class)

This class represents an instance of a dynamic list containing traffic events. It has flags for

which traffic event states to include, which are stronger than global filters as they cannot be cleared.

5.54.1.1.15TrafficEventDynListComparator (Class)

This class compares two TrafficEventDynListSubjects for the purposes of sorting.

5.54.1.1.16TrafficEventDynListSubject (Class)

This class represents a traffic event within the traffic event dynamic list.

5.54.1.1.17TrafficEventDynListSupporter (Class)

5.54.1.1.18VehiclesFilter (Class)

This class allows the event list to be filtered by the number of vehicles involved.

5.54.1.1.19WebObjectLocation Supporter (Class)

This interface allows common processing for objects supporting an ObjectLocation via the WebObjectLocation wrapper class..

5.54.1.1.20WebTrafficEvent (Class)

This class represents a TrafficEvent object in the system and caches its data for fast access. It provides accessor methods to get the cached data, in addition to auxiliary methods.

5.54.1.2 chartlite.servlet.trafficevents_classes (Class Diagram)

This diagram shows the various classes that are used to handle requests related to traffic events.

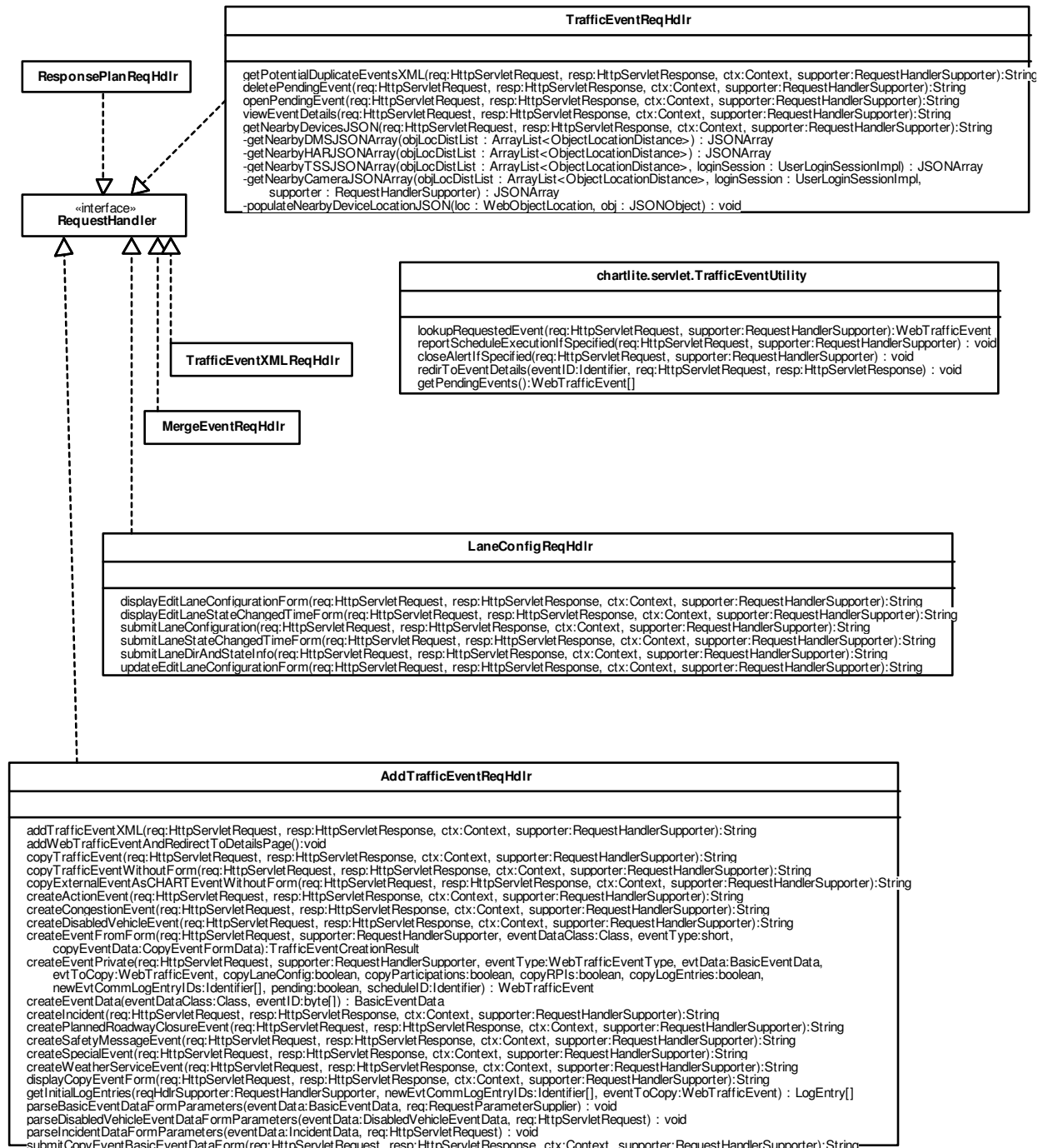


Figure 5-444. chartlite.servlet.trafficevents_classes (Class Diagram)

5.54.1.2.1 AddTrafficEventReqHdlr (Class)

This class is used to handle requests related to adding a traffic event to the system.

5.54.1.2.2 chartlite.servlet.TrafficEventUtility (Class)

This class contains methods that are useful for one or more traffic event related request handlers.

5.54.1.2.3 LaneConfigReqHdlr (Class)

This class handles any requests related to the traffic event lane configuration.

5.54.1.2.4 MergeEventReqHdlr (Class)

This class handles all requests related to merging traffic events.

5.54.1.2.5 RequestHandler (Class)

This interface specifies methods that are to be implemented by classes that are used to process requests.

5.54.1.2.6 ResponsePlanReqHdlr (Class)

This class handles requests related to traffic event response plans.

5.54.1.2.7 TrafficEventReqHdlr (Class)

This class handles requests related to traffic events that are not handled by one of the other specific traffic event request handlers.

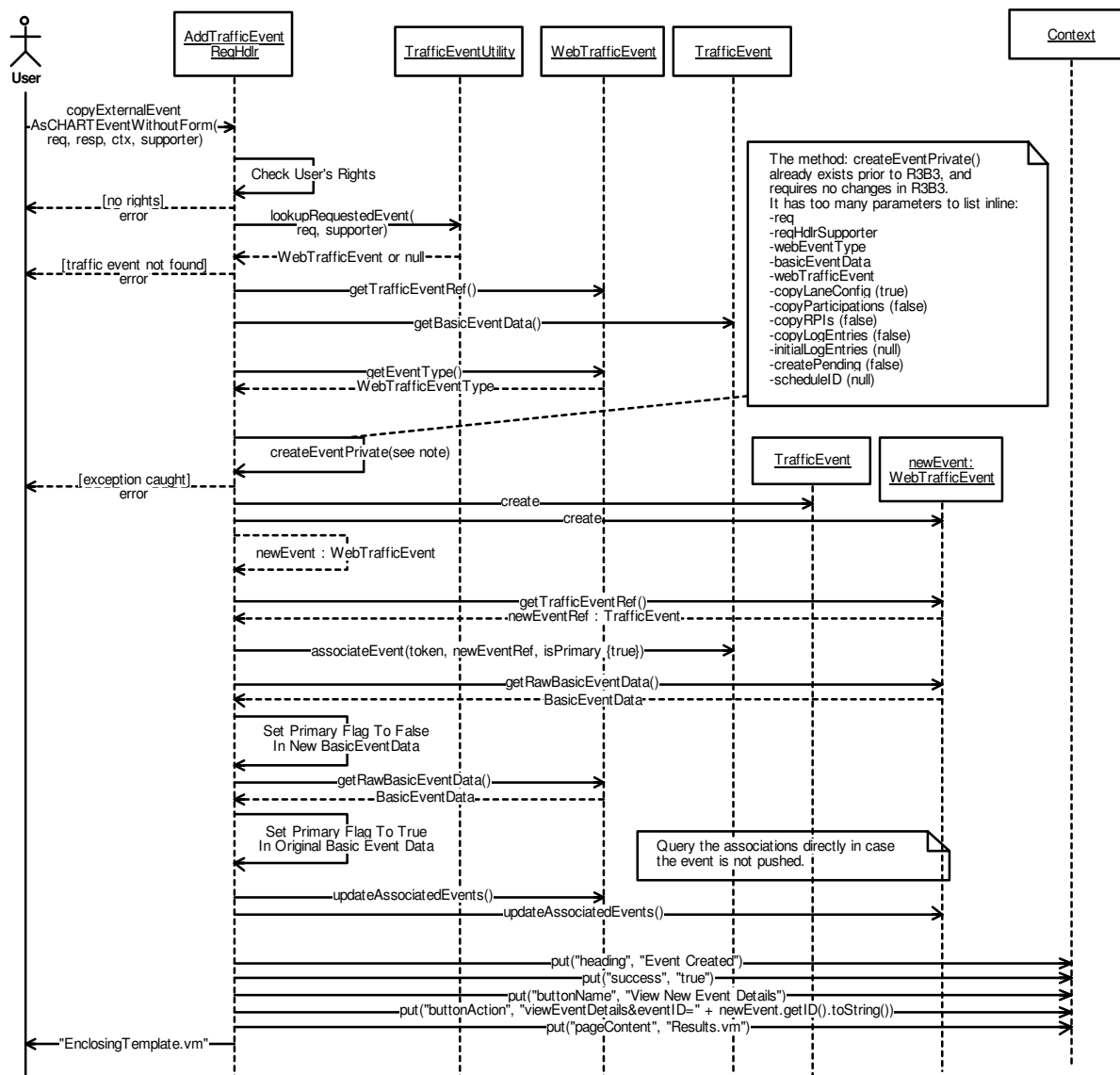
5.54.1.2.8 TrafficEventXMLReqHdlr (Class)

This class handles requests related to traffic events that return XML for the Flex2 application.

5.54.2 Sequence Diagrams

5.54.2.1 AddTrafficEventReqHdlr:copyExternalEventAsCHARTEventWithoutForm (Sequence Diagram)

This diagram shows how an external traffic event is copied as a CHART event. The traffic event with the requested ID is looked up in the cache. The existing createEventPrivate() method is called to create the copy of the traffic event and the WebTrafficEvent wrapper object for the new event. The original traffic event is then called to associate the new event, with the original being the primary traffic event. The cached traffic event data is updated for both events to reflect the primary/secondary flags, and the associations for both traffic events are queried from the server. The user is then shown the results page.



**Figure 5-445. AddTrafficEventReqHdlr:copyExternalEventAsCHARTEventWithoutForm
(Sequence Diagram)**

5.54.2.2 TrafficEventDynListSupporter:addCol (Sequence Diagram)

This diagram shows how a column is added to the list of columns to add to the TrafficEventDynList. If a sort property is specified, a TrafficEventDynListComparator is created. The DefaultDynListCol object is created to represent the column. The column is added to the list.

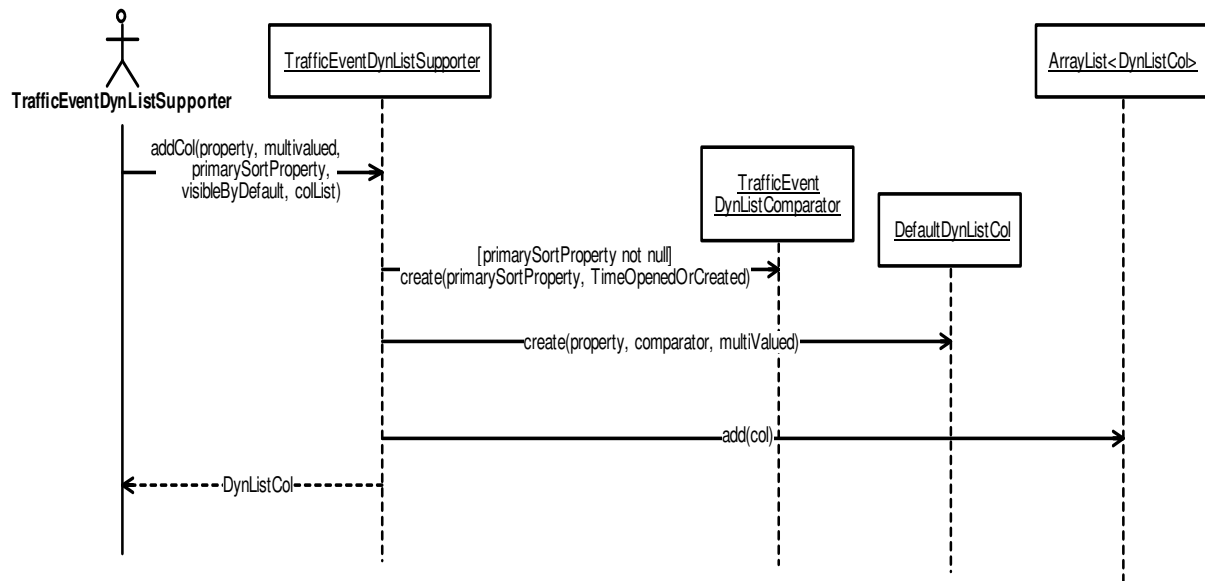


Figure 5-446. TrafficEventDynListSupporter:addCol (Sequence Diagram)

5.54.2.3 TrafficEventDynListSupporter:createDynList (Sequence Diagram)

This diagram shows how the traffic event dynamic list is created. First the column objects are created, as shown in the CreateDynListCols diagram. The TrafficEventDynList object is created. If the county/region/state is specified in the request, the filter value is set into the column. The event state inclusion flags are set based on request parameters indicating whether to display open, closed, pending, and/or external events. Finally the event type column is set as the default sort column.

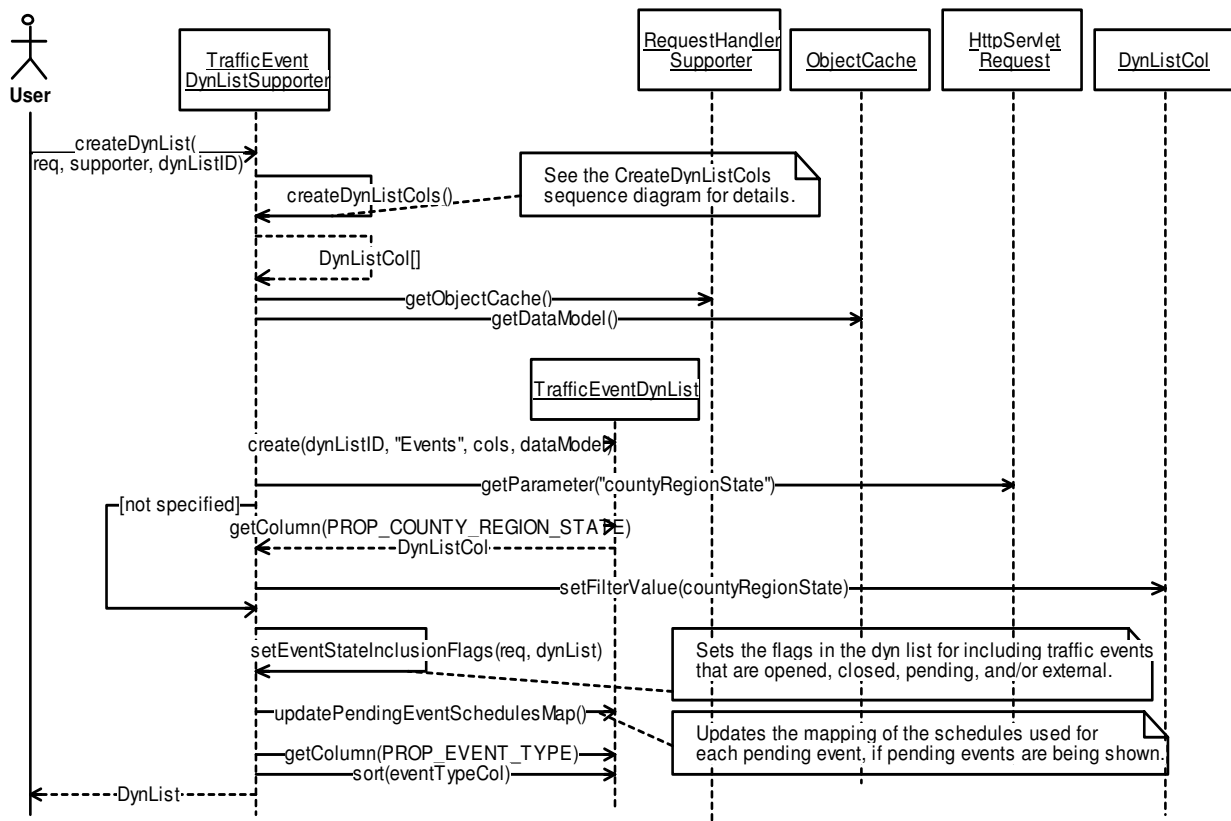


Figure 5-447. TrafficEventDynListSupporter:createDynList (Sequence Diagram)

5.54.2.4 TrafficEventDynListSupporter:createDynListCols (Sequence Diagram)

This diagram shows how the DynListCol objects are created to represent the columns of a traffic event dynamic list. The addCol() utility method is called to create the column and traffic event comparator, and add it to the newly created list. After creating a column, if a filter is appropriate for the column, the filter is created and set into the column. Finally the list is converted to an array and returned.

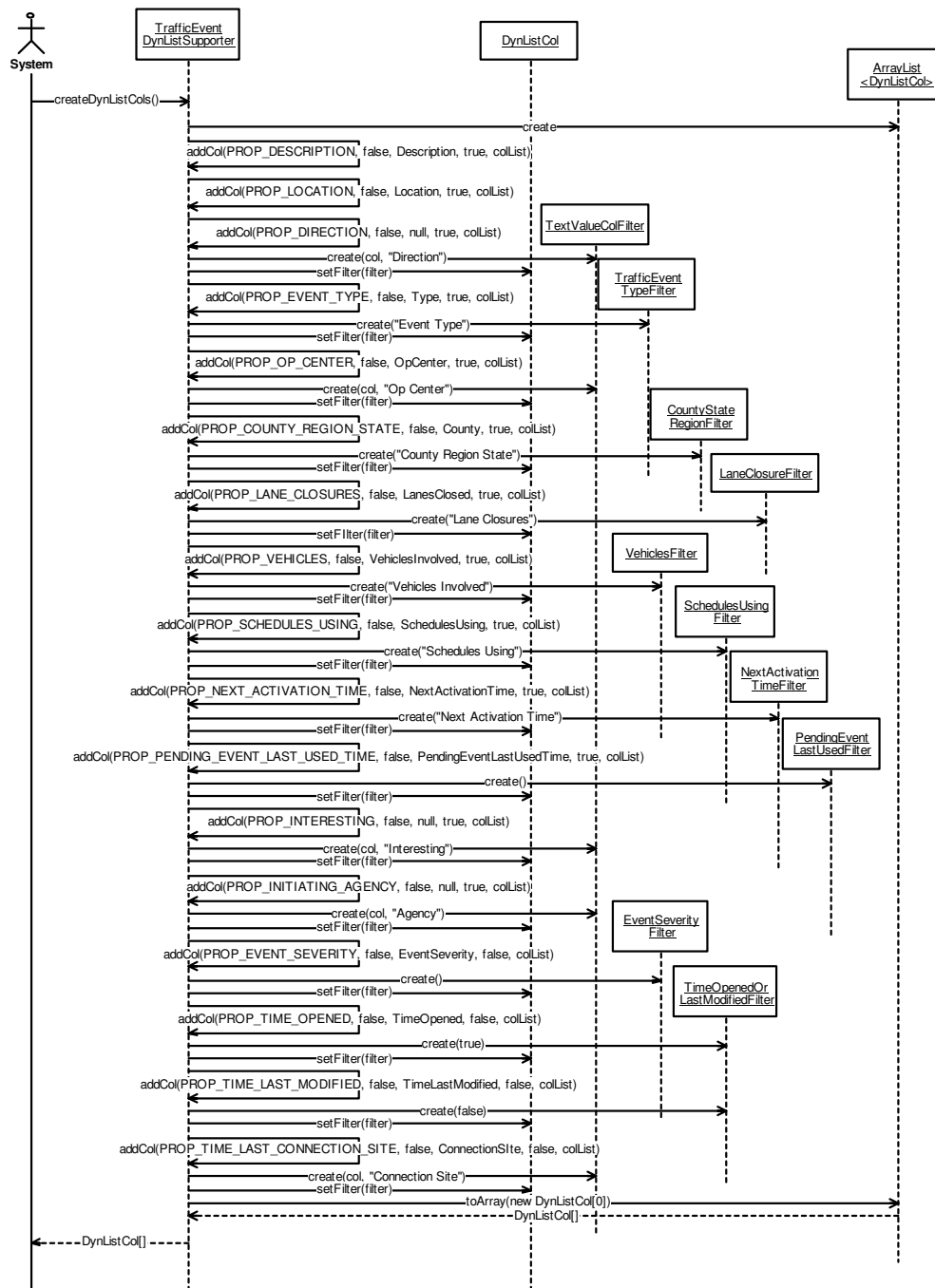


Figure 5-448. TrafficEventDynListSupporter:createDynListCols (Sequence Diagram)

5.54.2.5 TrafficEventReqHdlr:getNearbyCameraJSONArray (Sequence Diagram)

This diagram shows how the JSONArray object is built containing the information about nearby cameras. For each of the WebObjectLocationSupporter / distance pairs, the WebCamera and the distance are retrieved. A new JSON object is created, populated using information queried from the WebCamera (except the distance which was already calculated), and added to the JSONArray.

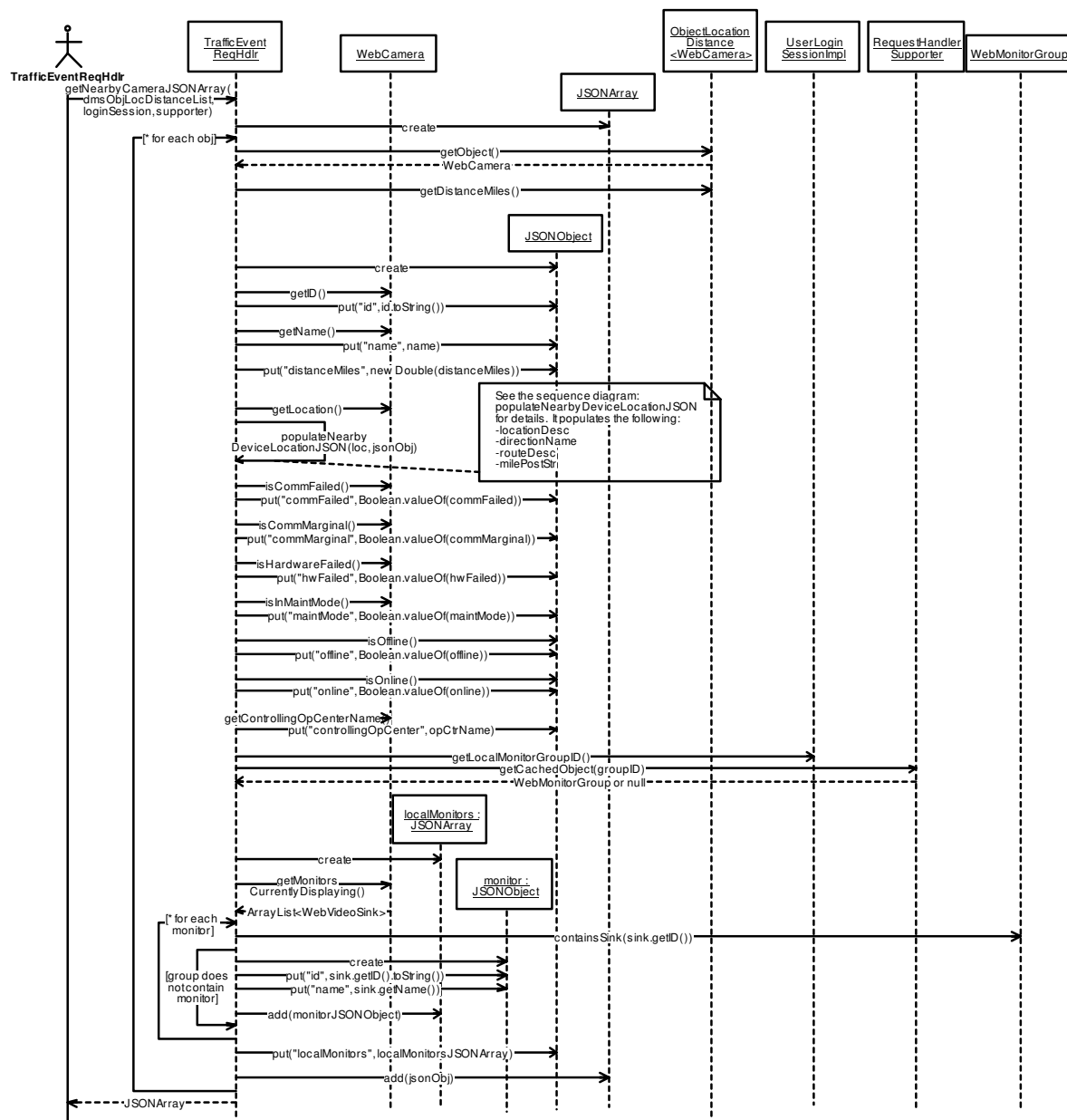


Figure 5-449. TrafficEventReqHdlr:getNearbyCameraJSONArray (Sequence Diagram)

5.54.2.6 TrafficEventReqHdlr:getNearbyDMSJSONArray (Sequence Diagram)

This diagram shows how the JSONArray object is built containing the information about nearby DMSs. For each of the WebObjectLocationSupporter / distance pairs, the WebDMS and the distance are retrieved. A new JSON object is created, populated using information queried from the WebDMS (except the distance which was already calculated), and added to the JSONArray.

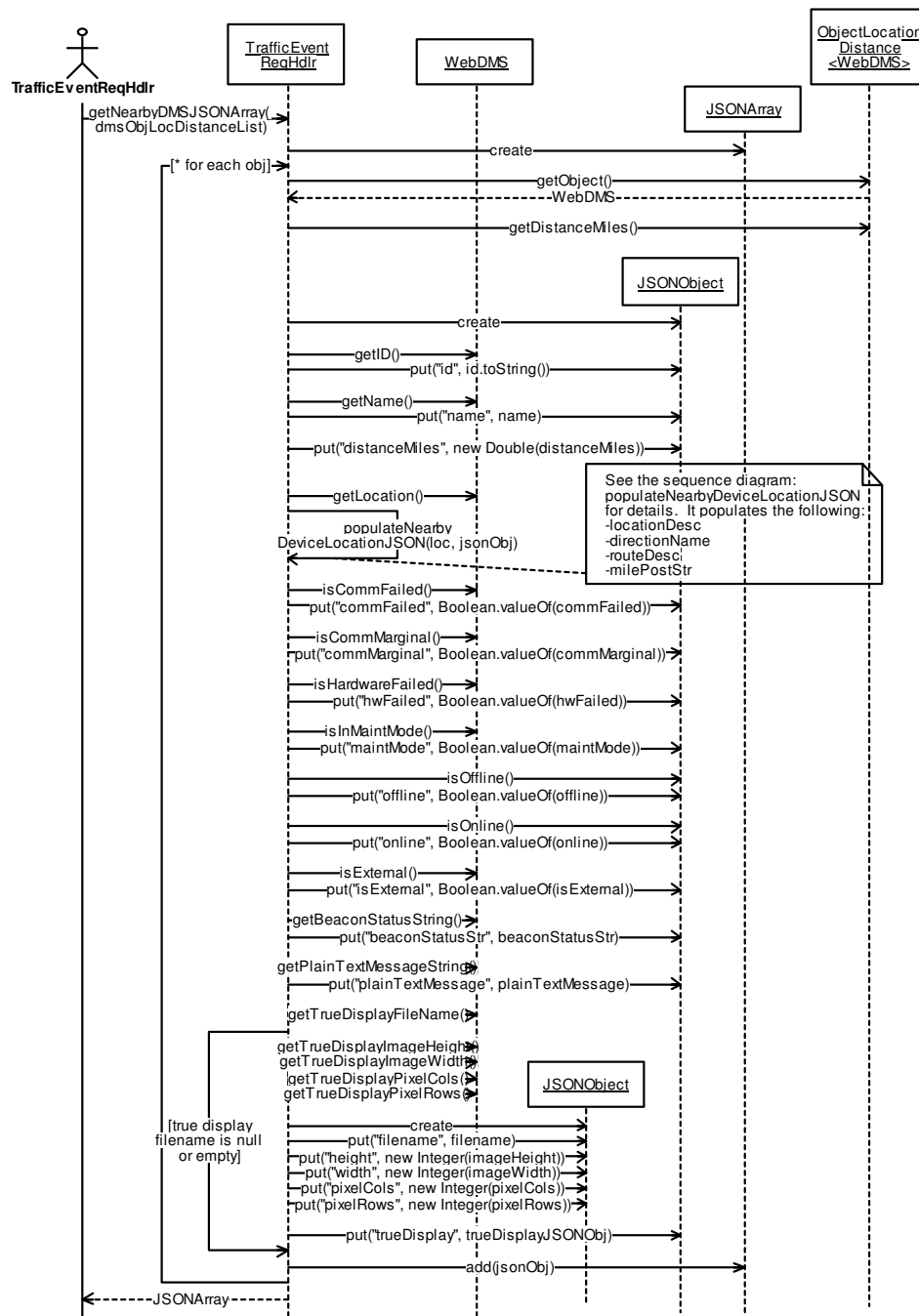


Figure 5-450. TrafficEventReqHdr::getNearbyDMSJSONArray (Sequence Diagram)

5.54.2.7 TrafficEventReqHdlr:getNearbyDevicesJSON (Sequence Diagram)

This diagram shows the processing to find the devices near a traffic event. The event ID in the request is used to look up the WebTrafficEvent object from the cache. Its GeoLocation object is queried, which may be null if the traffic event does not have a defined location. The WebDevice objects are queried from the cache, and split up into lists for DMS, HAR, TSS, and Camera. External DMSs and TSSs will be excluded if the user does not have rights to view them. Next, each list is passed to ServletUtil.getNearbyDevices() to get an ordered list of devices within the distance retrieved from the login session. The returned list is a subset of the list passed in (see the ServletUtil:getNearbyDevices sequence diagram for details). Next, each returned list is passed to a private method to generate a JSONArray object which contains a list of JSONObject objects representing each type of device, with device-specific fields. These arrays (or an error message) are put into a root JSON object which is sent back in the response. The web page will have a handler to use the data to dynamically alter the nearby device list.

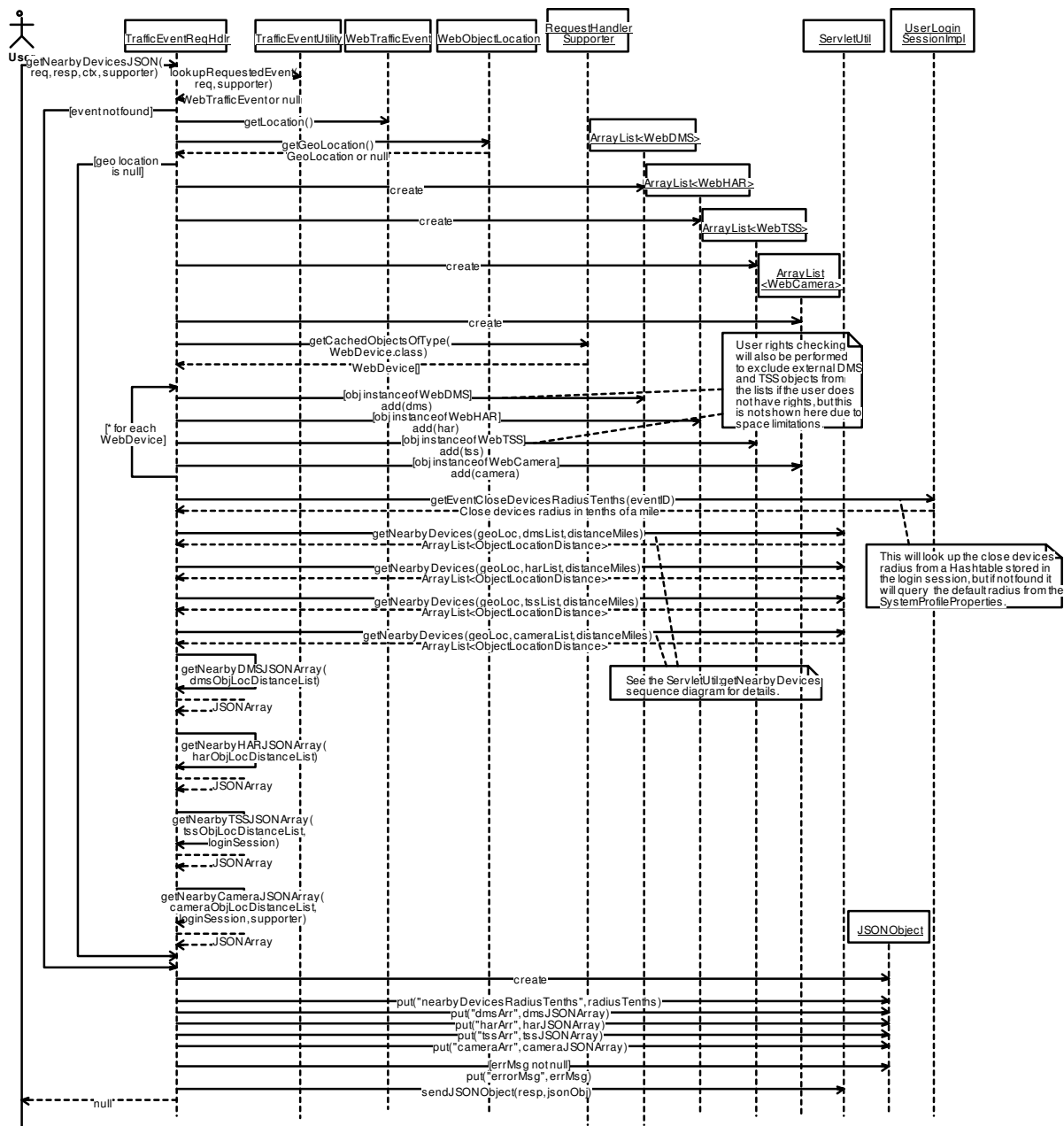


Figure 5-451. TrafficEventReqHdlr: getNearbyDevicesJSON (Sequence Diagram)

5.54.2.8 TrafficEventReqHdlr:getNearbyHARJSONArray (Sequence Diagram)

This diagram shows how the JSONArray object is built containing the information about nearby HARs. For each of the WebObjectLocationSupporter / distance pairs, the WebHAR and the distance are retrieved. A new JSON object is created, populated using information queried from the WebHAR (except the distance which was already calculated), and added to the JSONArray.

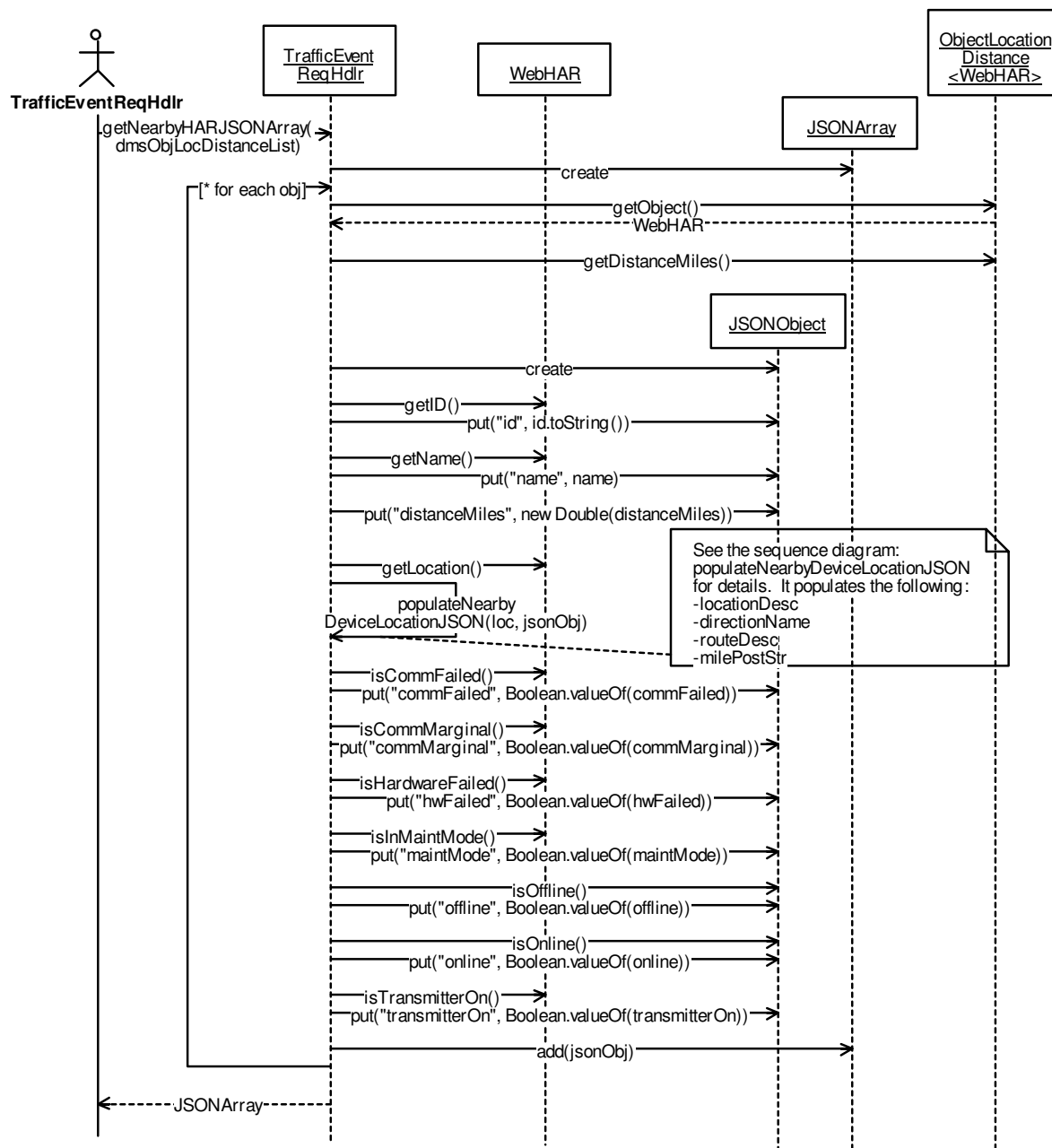


Figure 5-452. TrafficEventReqHdlr:getNearbyHARJSONArray (Sequence Diagram)

5.54.2.9 TrafficEventReqHdlr:getNearbyTSSJSONArray (Sequence Diagram)

This diagram shows how the JSONArray object is built containing the information about nearby TSSs. For each of the WebObjectLocationSupporter / distance pairs, the WebTSS and the distance are retrieved. A new JSON object is created, populated using information queried from the WebTSS (except the distance which was already calculated), and added to the JSONArray.

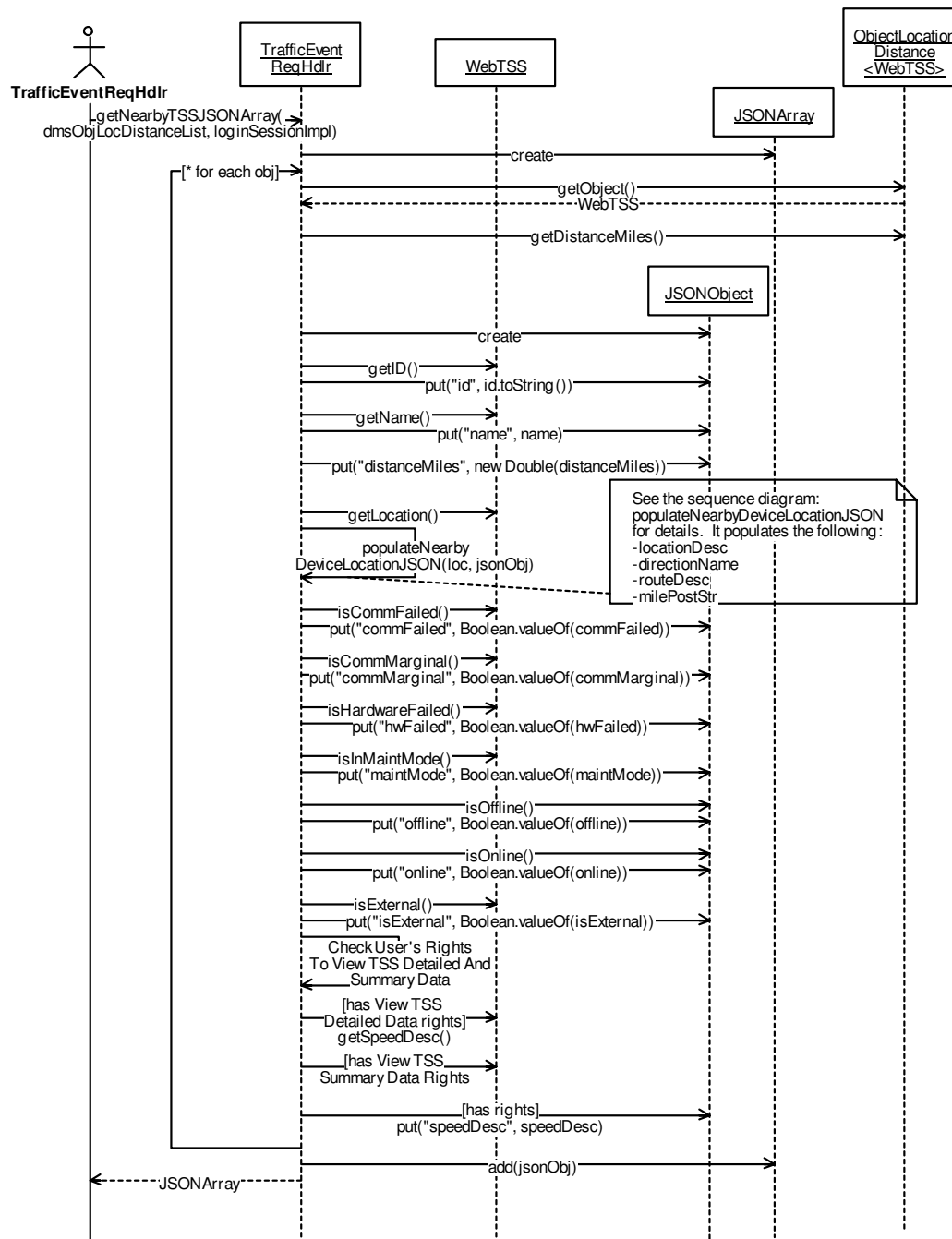
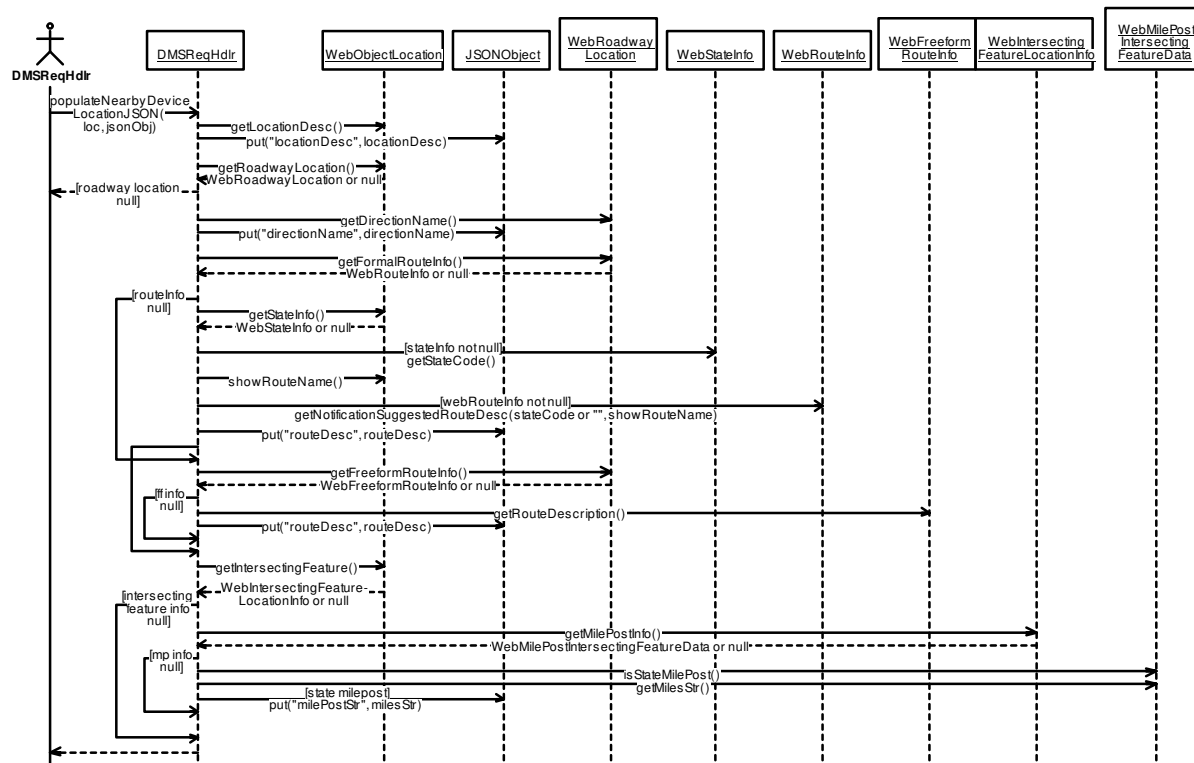


Figure 5-453. TrafficEventReqHdr:getNearbyTSSJSONArray (Sequence Diagram)

5.54.2.10 TrafficEventReqHdlr:populateNearbyDeviceLocationJSON (Sequence Diagram)

This diagram shows how location-related parameters are populated in the JSONObject for a nearby device, given the WebObjectLocation for the devices. The parameters populated are locationDesc, directionName, routeDesc, and milePostStr.



5-454. TrafficEventReqHdlr:populateNearbyDeviceLocationJSON (Sequence Diagram)

5.54.2.11 TrafficEventReqHdlr:setNearbyDevicesRadiusJSON (Sequence Diagram)

This diagram shows the processing when the user changes the nearby device radius for a traffic event. The event ID and the distance in tenths of a mile are parsed from the request parameters. The user login session is then called to save the radius for the traffic event, which will preserve it as long as the user is logged in. Finally the getNearbyDevicesJSON() method is called directly to return the nearby devices for the new distance value. See the getNearbyDevicesJSON sequence diagram for details.

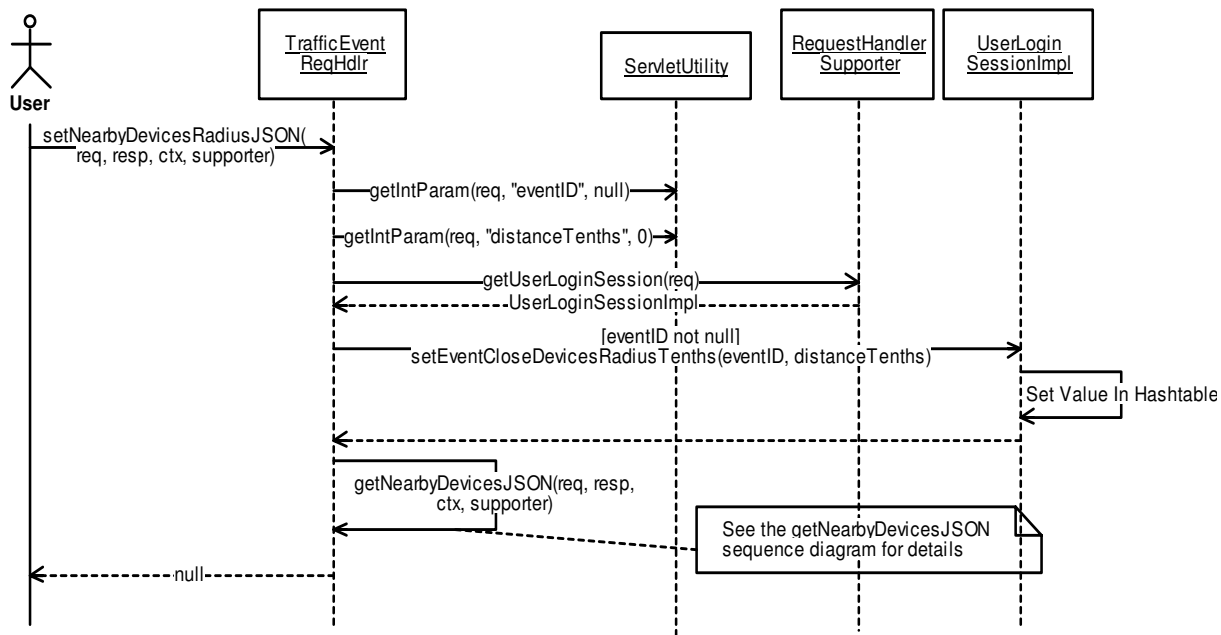


Figure 5-455. TrafficEventReqHdlr:setNearbyDevicesRadiusJSON (Sequence Diagram)

5.54.2.12 TrafficEventReqHdlr:viewEventDetails (Sequence Diagram)

This diagram shows the processing to view the event details page. The traffic event with the requested ID is retrieved from the cache and put into the Velocity context. (The logic is unchanged for R3B2, and is only reproduced here to explain that the Velocity template will extract the notification records from this traffic event object by calling `getNotificationRecords()`.) The lane configuration is obtained from the traffic event and is used to get the lane display GIF file name and metadata, and those are put into the Velocity context. The response devices are retrieved and are put into the context, as are the road conditions and the event types (for copy event). The recently viewed events list is updated within the `UserLoginSession`. Finally the template name is returned so it can be processed by Velocity.

In R3B3, additional user rights checking is added to the template to display either the actual or cleansed event name, actual or cleansed incident type (if applicable), and to show or hide the event history links.

5.54.2.13 TrafficEventXMLReqHdr:getOpenTrafficEventsXML (Sequence Diagram)

This diagram shows the processing that takes place to populate the list of traffic events shown on the user's home page. This processing is invoked by the home page Flex2 application when the home page is first loaded, and periodically to refresh the traffic event information. The Flex2 application uses the XML returned from this request to display the traffic events for which the operation center is responsible. This is done on a tab control, with the events filtered onto different tabs based on the event type. Each tab shows the number of events of that type that appear on the tab, allowing the user to know the number of open events of each type even for tabs that are not the currently displayed tab.

For each event that appears on a tab, the following information will be shown: - The event name, which is also a link that when clicked shows the event details page in the user's working window - The event location - The county/state (if specified in the traffic event)

Lane closure information will be shown for Incidents, planned closures, and special events if lane closure data has been specified in the event.

If the event is an incident, the tab will also show the the vehicles involved information if this data has been specified for the traffic event.

If the event is a congestion event, the tab will show the "recurring" indicator if the event has been flagged as recurring congestion.

If the event is a disabled vehicle event, the tab will contain the color/make and/or tag information if that data has been entered into the traffic event.

If the event is a weather service event, road condition data will be shown if it has been specified in the event.

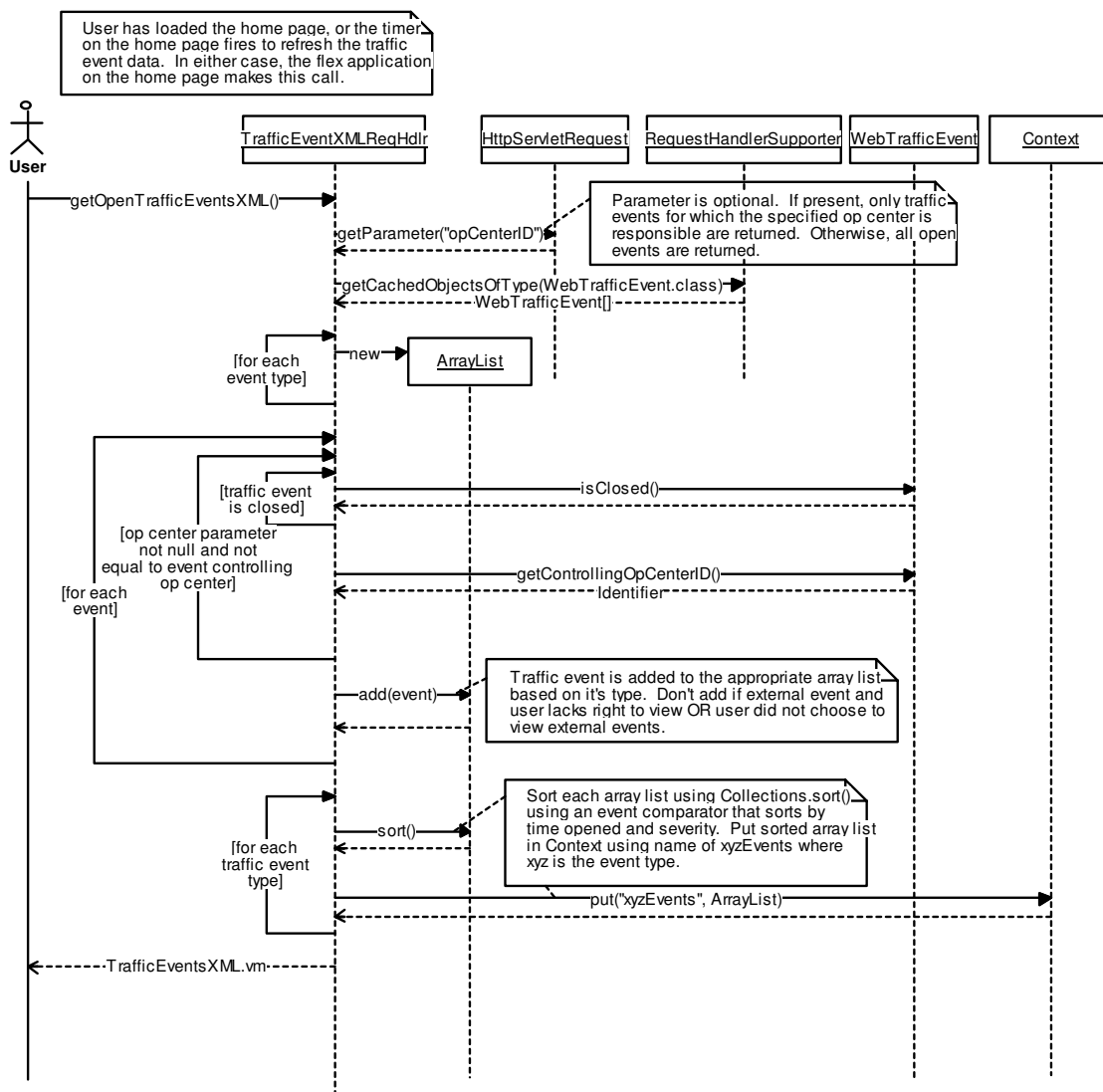


Figure 5-457. TrafficEventXMLReqHdlr:getOpenTrafficEventsXML (Sequence Diagram)

5.55 Chartlite.servlet.trafficevents.location

5.55.1 Classes

5.55.1.1 chartlite.servlet.location_classes (Class Diagram)

This diagram shows CHART GUI servlet classes related to locations.

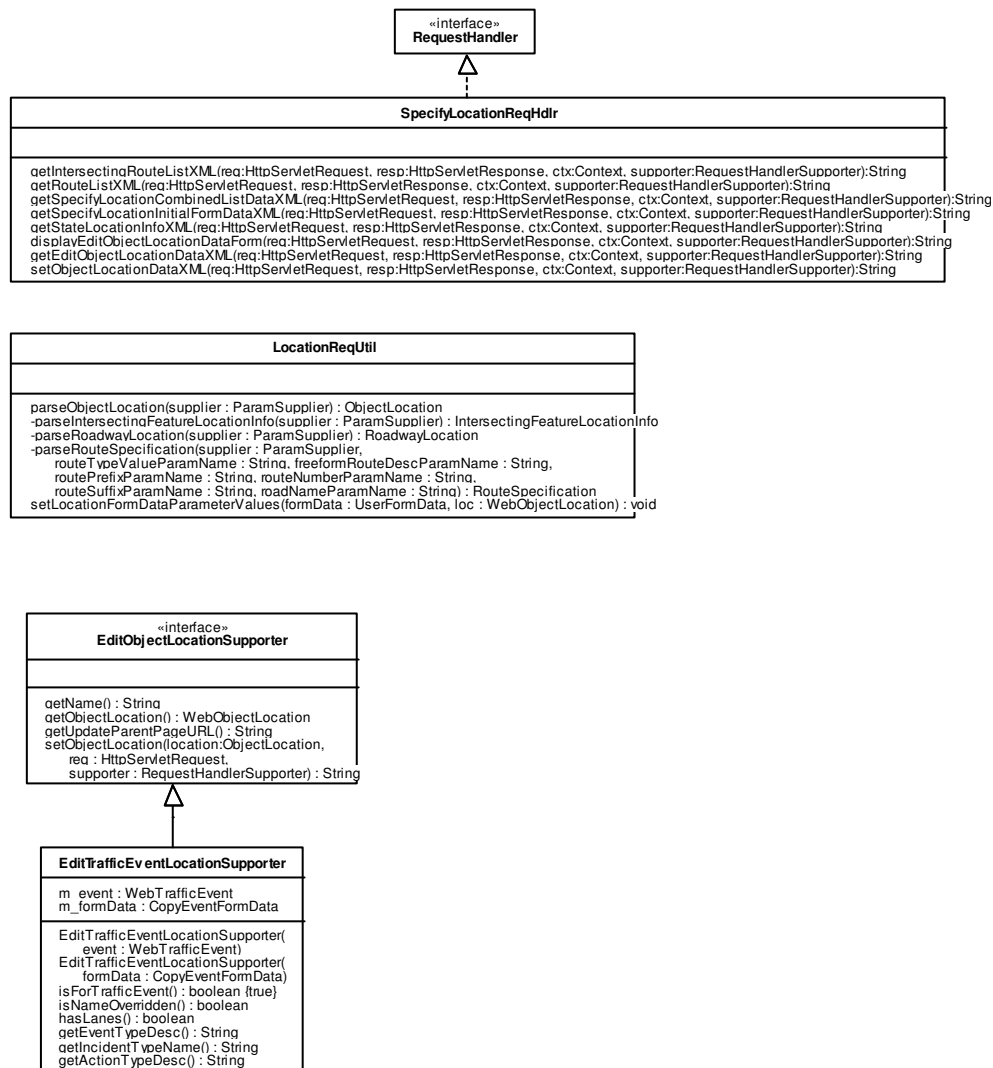


Figure 5-458. chartlite.servlet.location_classes (Class Diagram)

5.55.1.1.1 EditObjectLocationSupporter (Class)

This interface provides functionality allowing the location data to be edited. (For example, the target of the edited location may be an existing object, or it may be a form data object for creating a new object).

5.55.1.1.2 EditTrafficEventLocationSupporter (Class)

This class provides functionality for editing the location of an existing traffic event, or one that is being copied.

5.55.1.1.3 LocationReqUtil (Class)

This class provides functionality for location-related requests.

5.55.1.1.4 RequestHandler (Class)

This interface specifies methods that are to be implemented by classes that are used to process requests.

5.55.1.1.5 SpecifyLocationReqHdlr (Class)

5.55.2 Sequence Diagrams

5.55.2.1 SpecifyLocationReqHdlr:displayEditObjectLocationDataForm (Sequence Diagram)

This diagram shows the processing to display the Edit Object Location Data form. This diagram requires that the type-specific part of the code performs the following preconditions: 1) check the user rights (if appropriate), 2) create an object implementing the EditObjectLocationSupporter interface, 3) put the object in the TempObjectStore, and 4) redirect the response to this request using the "editLocationSupporterID" parameter. This request will retrieve the EditObjectLocationSupporter object from the TempObjectStore and will put it in the Velocity context and load the Edit Object Location Data Form template.

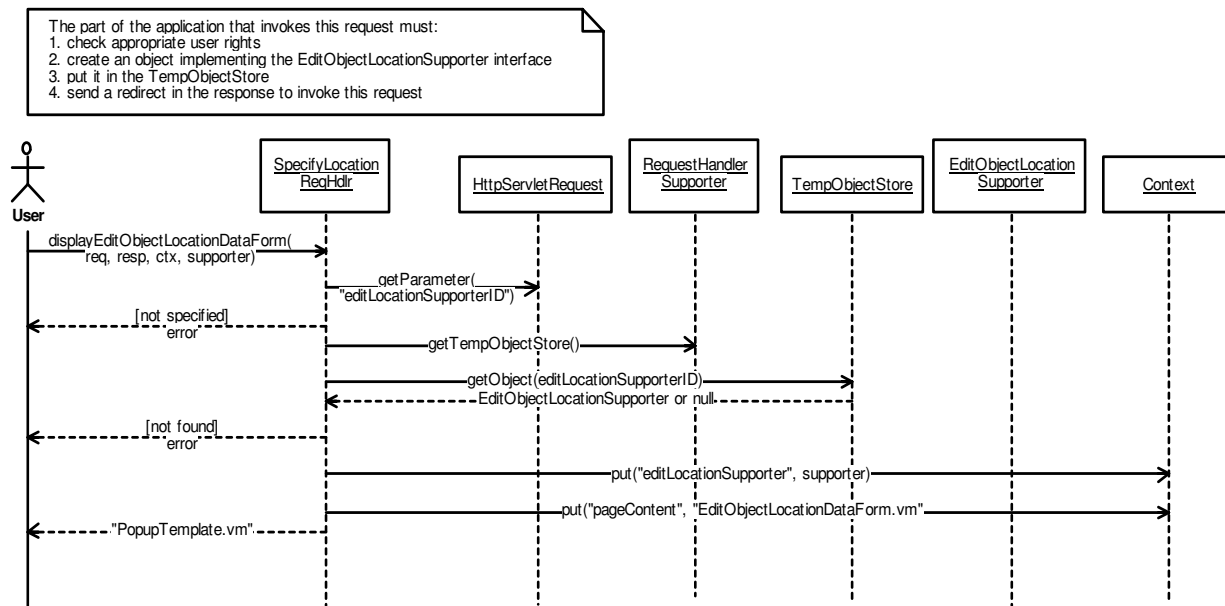


Figure 5-459. SpecifyLocationReqHdlr:displayEditObjectLocationDataForm (Sequence Diagram)

5.55.2.2 SpecifyLocationReqHdlr:getEditObjectLocationDataXML (Sequence Diagram)

This diagram shows how the location data XML is built for populating the Edit Location Flex application. The ID of the edit location supporter object is retrieved and is used to find the object in the TempObjectStore. (This should not fail because the request was invoked by the Flex control, which also needed the supporter object.) This supporter object is put into the Velocity context and the EditObjectLocationDataXML template is returned. The template will query the location data (and the extra data for traffic events) to populate the XML that is being returned.

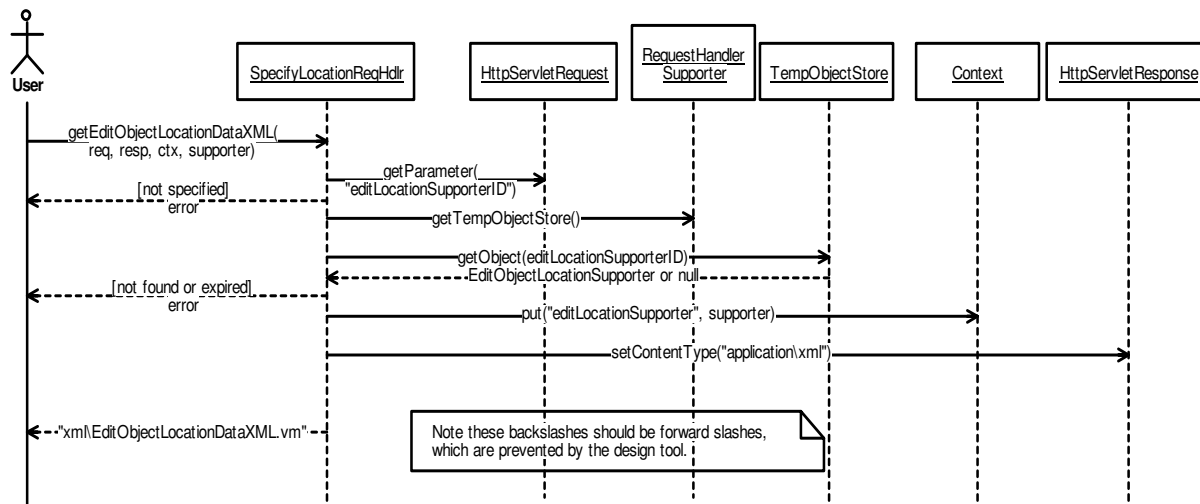


Figure 5-460. SpecifyLocationReqHdlr:getEditObjectLocationDataXML (Sequence Diagram)

5.55.2.3 SpecifyLocationReqHdlr:setObjectLocationDataXML (Sequence Diagram)

This diagram shows the processing when the Edit Object Location Data Form is submitted. The edit location supporter ID from the request is used to retrieve the object from the TempObjectStore. The ObjectLocation is parsed from the request parameters, and the EditObjectLocationSupporter is called to set the location. The details of this will be specific to the type of object, and are not shown here. The XML result indicating success or failure is returned.

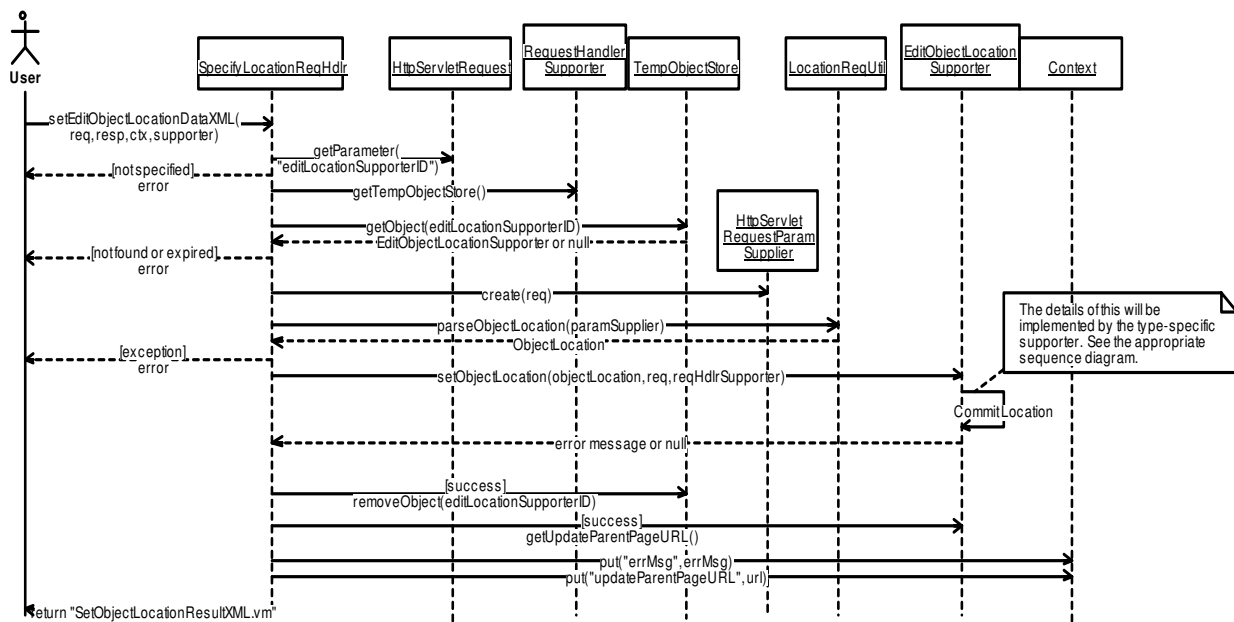


Figure 5-461. SpecifyLocationReqHdlr:setObjectLocationDataXML (Sequence Diagram)

5.56 Chartlite.servlet.shazam

5.56.1 Class Diagrams

5.56.1.1 GUIHAZAMServletClasses (Class Diagram)

This diagram shows classes used by the servlet to process requests related to SHAZAM devices.

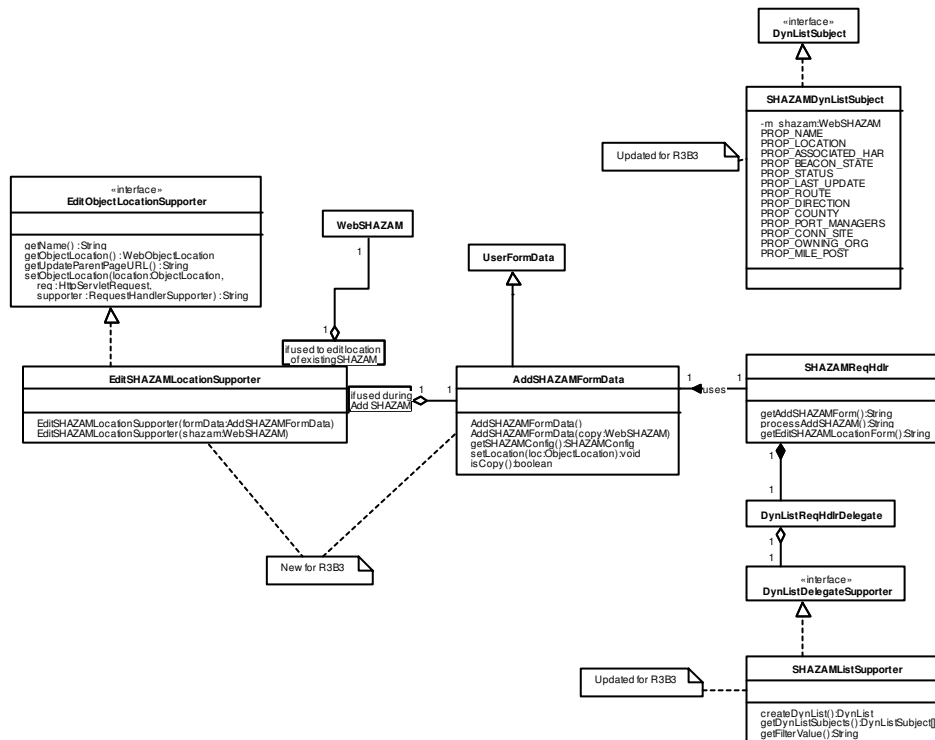


Figure 5-462. GUIHAZAMServletClasses (Class Diagram)

5.56.1.1.1 AddSHAZAMFormData (Class)

This class is used to store data from the Add SHAZAM form while interim pages are displayed (such as the edit location form).

5.56.1.1.2 DynListDelegateSupporter (Class)

This interface contains functionality to support the DynListReqHdrDelegate

5.56.1.1.3 DynListReqHdlrDelegate (Class)

This class helps request handlers support dynamic lists. Requests to view, sort, or filter dynamic lists can be passed from a request handler to this class, provided the URL used for the requests contain parameters required by this class, such as the id of the list, the property name, and/or the filter value.

5.56.1.1.4 DynListSubject (Class)

This interface is implemented by classes that wish to be capable of being displayed in a dynamic list.

5.56.1.1.5 EditObjectLocationSupporter (Class)

This interface provides functionality allowing the location data to be edited. (For example, the target of the edited location may be an existing object, or it may be a form data object for creating a new object).

5.56.1.1.6 EditSHAZAMLocationSupporter (Class)

This class is used to support the generic edit location operation for SHAZAM devices.

5.56.1.1.7 SHAZAMDynListSubject (Class)

This class is a wrapper for a WebSHAZAM object that allows it to be displayed in a dynamic list.

5.56.1.1.8 SHAZAMListSupporter (Class)

This class is a DynListDelegateSupporter for the SHAZAM dynamic list. It provides SHAZAM specific functionality to the generic DynListReqHdlrDelegate.

5.56.1.1.9 SHAZAMReqHdlr (Class)

This class processes requests related to SHAZAM devices.

5.56.1.1.10 UserFormData (Class)

This class is used to store form data between requests while a user is editing a complex form, and provides convenience methods for parsing the values from the request.

5.56.1.1.11 WebSHAZAM (Class)

This class is a wrapper for a SHAZAM CORBA object, used to cache data related to the SHAZAM in the GUI object cache and to provide access to the SHAZAM configuration and status data on web pages.

5.56.2 Sequence Diagrams

5.56.2.1 EditSHAZAMLocationSupporter:setObjectLocation (Sequence Diagram)

This diagram shows the processing that is performed when the EditSHAZAMLocationSupporter's setObjectLocation method is called. This method is called by the generic location editing request handler when the user submits the edit location form. If the supporter was constructed using a form data object (which is the case if an Add SHAZAM operation is in progress), the location data stored inside the form data is updated and the method returns. If the supporter was constructed using a WebSHAZAM, its CORBA object reference is called to set the location data in the CORBA SHAZAM object. The location data in the WebSHAZAM is updated and the method returns.

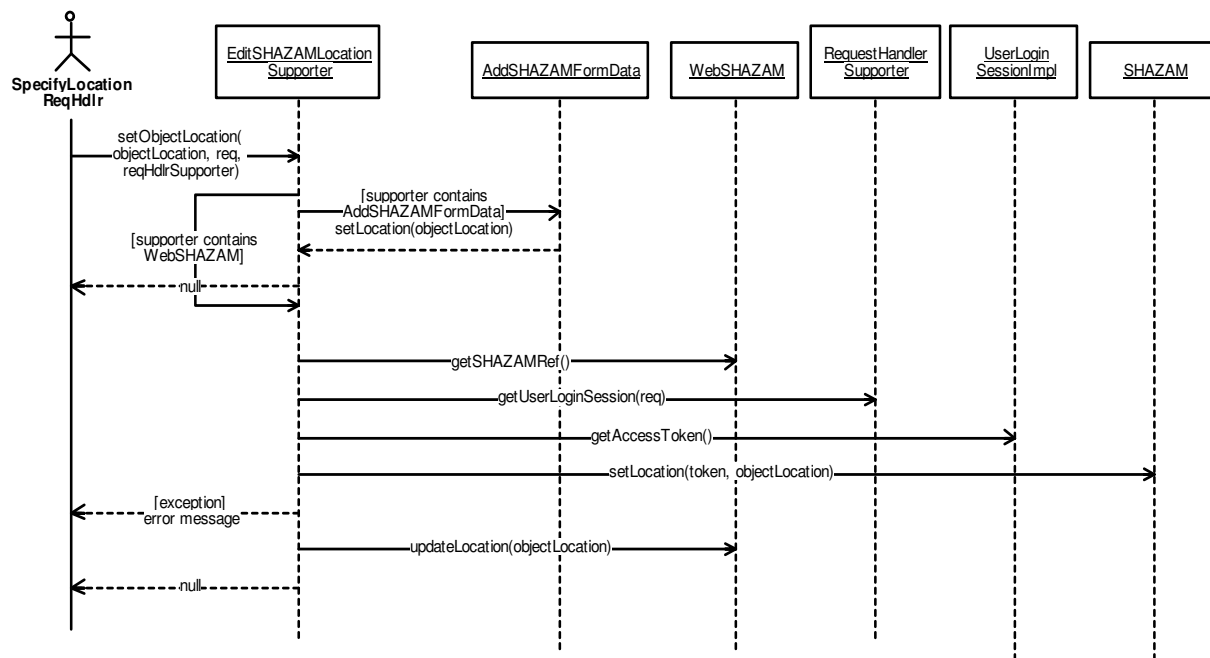


Figure 5-463. EditSHAZAMLocationSupporter:setObjectLocation (Sequence Diagram)

5.56.2.2 SHAZAMListSupporter:createDynList (Sequence Diagram)

This diagram shows the processing that is performed when the SHAZAMListSupporter is called to create a SHAZAM dynamic list. A DefaultDynListCol object is created for each column and stored in an ArrayList. A derivation of DynListComparator is constructed and added to each column, and a BaseDynListFilter derived object is also added to the column if the column supports filtering. The list of columns is used to construct a DefaultDynList. The filterType parameter (if present) is used to make other filter related configurations to the dyn list. If the filterType parameter is set to "OpCenterFolders", an OpCenterFolderFilter is constructed, configured for the user's op center, and added to the dyn list as a global filter which will show only SHAZAMs that exist in folders that are tagged for use with the user's op center. Other filterType values are used to set the status filter, and in the case of the "OnlineActivated" filter type, configuration for the beacon filter.

This processing is updated in R3B3 to add support for the following columns: Route, Direction, County, Port Managers, Connection Site, Owning Org, Mile Post.

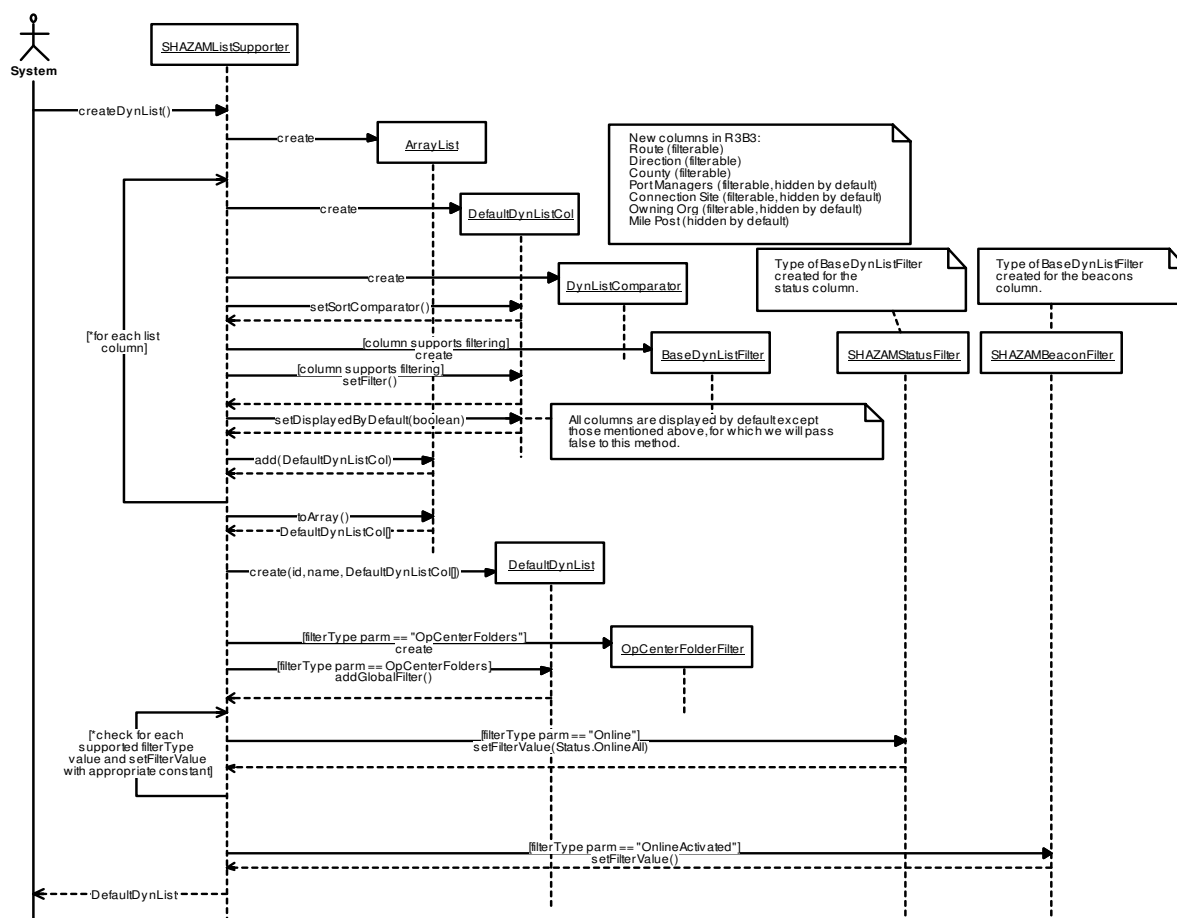


Figure 5-464. SHAZAMListSupporter:createDynList (Sequence Diagram)

5.56.2.3 SHAZAMReqHdlr:getAddSHAZAMForm (Sequence Diagram)

NOTE: getAddSHAZAMForm exists prior to R3B3 and is modified in R3B3 to use a form data object to support the multi-page entry process that includes setting the SHAZAM location.

This diagram shows the processing that is performed when the administrator chooses to add a SHAZAM to the system or create a new SHAZAM by copying an existing SHAZAM. This processing is also performed if an add or copy is already in progress and the Add form is being redisplayed after navigating away from the add form to set the location data. If this is the case, a formID parameter will be used to retrieve the form data object from the temp object store. If performing a copy, the WebSHAZAM being copied will be retrieved from the object cache and will be used to create an AddSHAZAMFormData object. This will have the effect of pre-populating the Add SHAZAM form with data from the SHAZAM being copied. If this is an Add SHAZAM operation that was not already in progress (neither formID nor shazamID is present in the request), a new empty AddSHAZAMFormData object is created. The form data is stored in the temp object store if not already stored there. Other objects needed for the form for select lists are obtained from the object cache and placed in the context. The Add SHAZAM form is then displayed to the user.

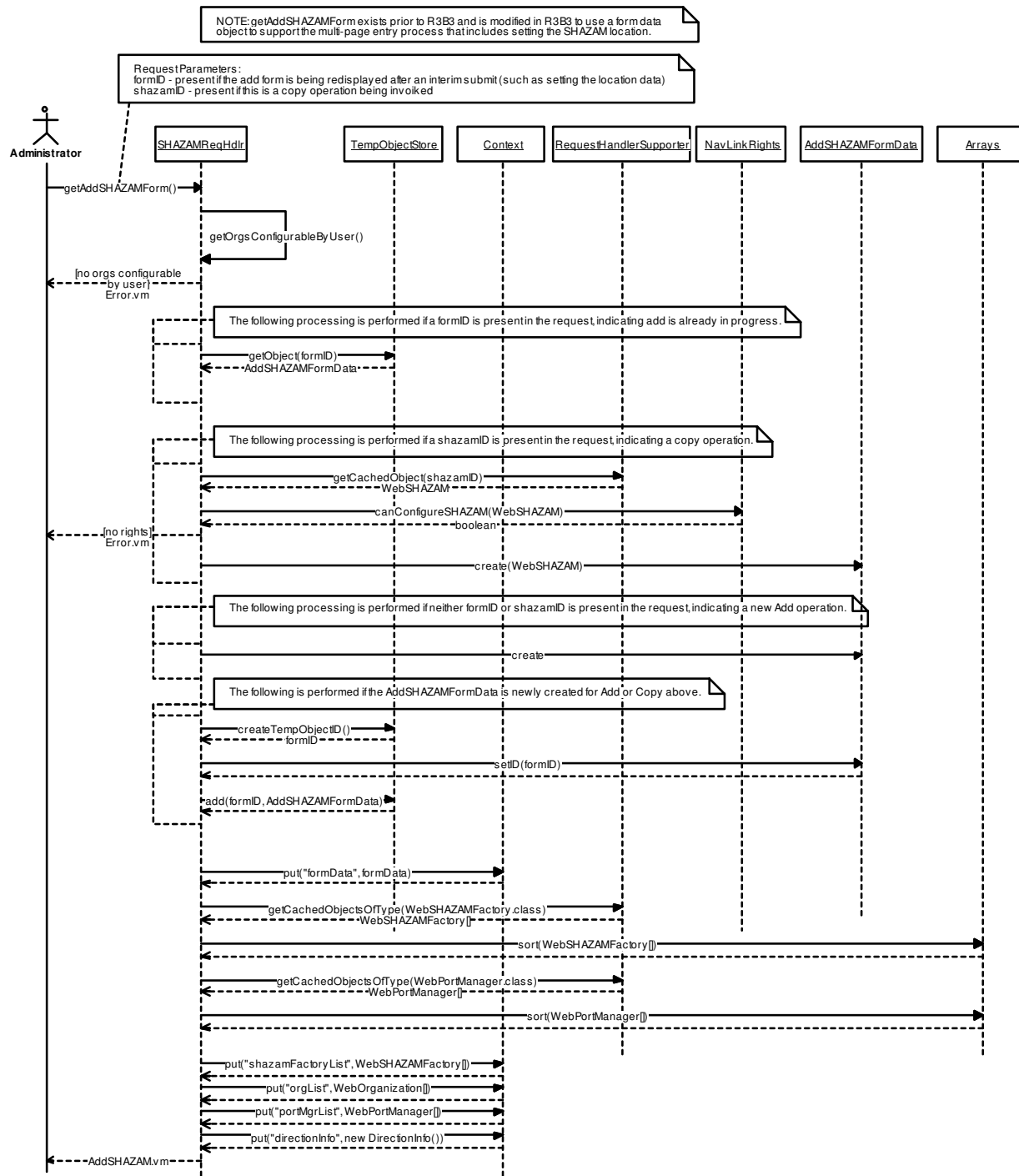


Figure 5-465. SHAZAMReqHdr: getAddSHAZAMForm (Sequence Diagram)

5.56.2.4 SHAZAMReqHdlr:getSHAZAMEditLocationForm (Sequence Diagram)

This diagram shows the processing that is performed when the user chooses to edit the location fields for a SHAZAM. This can be done as part of the Add SHAZAM process, or can be done on an existing SHAZAM from its details page. If done during the Add SHAZAM process, a formID parameter will be present and is used to retrieve the AddSHAZAMFormData object from the temp object store. The populateFromRequest method is called to store any data the user had entered in fields of the Add SHAZAM form prior to choosing to set the location fields. An EditSHAZAMLocationSupporter object is then created, passing the AddSHAZAMFormData to the constructor.

If a shazamID is passed in the request, this indicates the operation is being performed "stand alone" from the SHAZAM details page. In this case, the user rights must be checked, and the WebSHAZAM is retrieved from the object cache. An EditSHAZAMLocationSupporter is then constructed from the WebSHAZAM object.

After the EditSHAZAMLocationSupporter is created (either with AddSHAZAMFormData or a WebSHAZAM), it is placed in the temp object store so it can be accessed by the generic location editing request handler. The request is then redirected to the request that shows the generic edit location form.

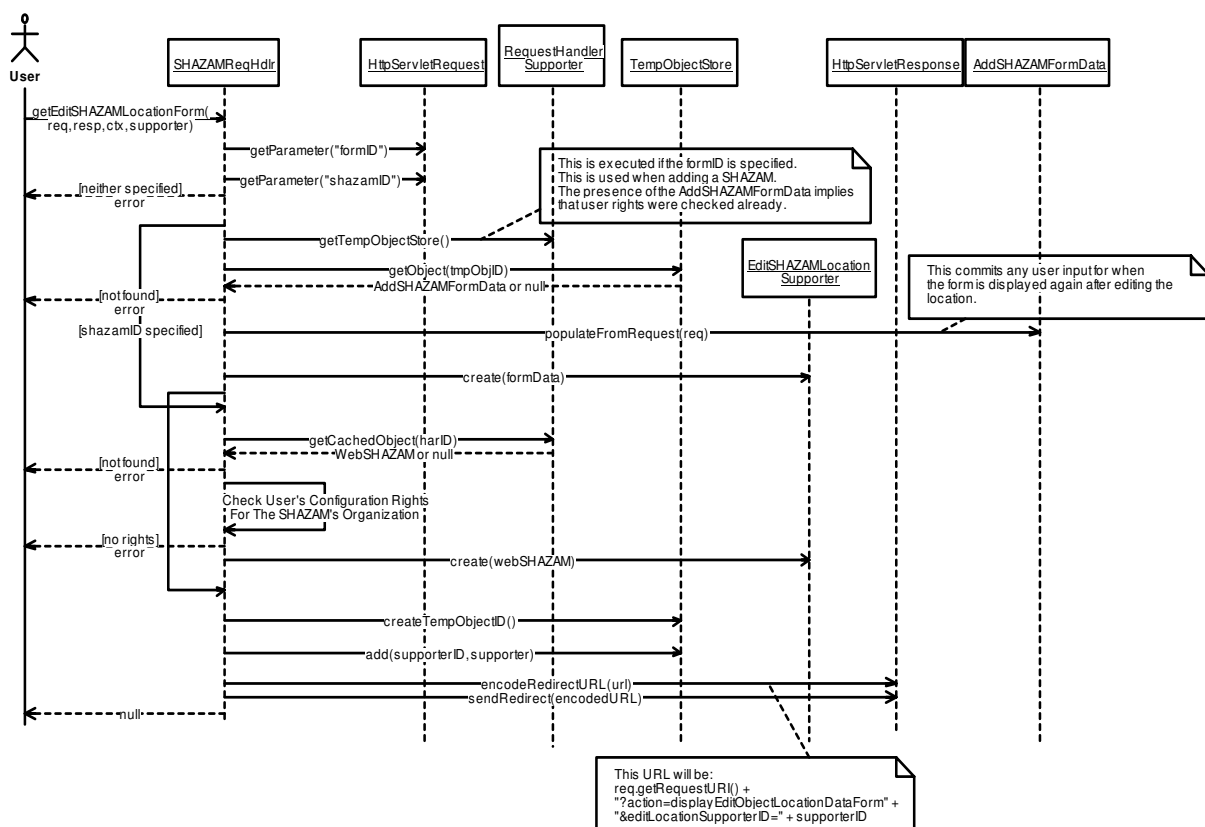


Figure 5-466. SHAZAMReqHdlr:getSHAZAMEditLocationForm (Sequence Diagram)

5.56.2.5 SHAZAMReqHdlr:processAddSHAZAM (Sequence Diagram)

NOTE: processAddSHAZAM exists prior to R3B3 and is modified in R3B3 to use a form data object to support the multi-page entry process that includes setting the SHAZAM location.

This diagram shows the processing that takes place when the user submits the Add SHAZAM Form. The formID is retrieved from the request parameters and is used to retrieve the form data from the temp object store. The form data's populateFromRequest method is used to get all request parameters into the form data object, and its getSHAZAMConfig() method is used to populate a SHAZAMConfig object using the request parameters. The selected creation site (factory) is not part of the SHAZAMConfig, so it is retrieved from the form data separately. Any errors that are detected by the form data when creating the SHAZAMConfiguration or when attempting to get the selected factory ID cause the AddSHAZAM form to be redisplayed with an error message. If there were no errors, the SHAZAMConfiguration is passed to the SHAZAMFactory to create a new SHAZAM object. The new SHAZAM is then called to obtain its ID, status, and configuration to allow a WebSHAZAM object to be created and stored in the GUI cache.

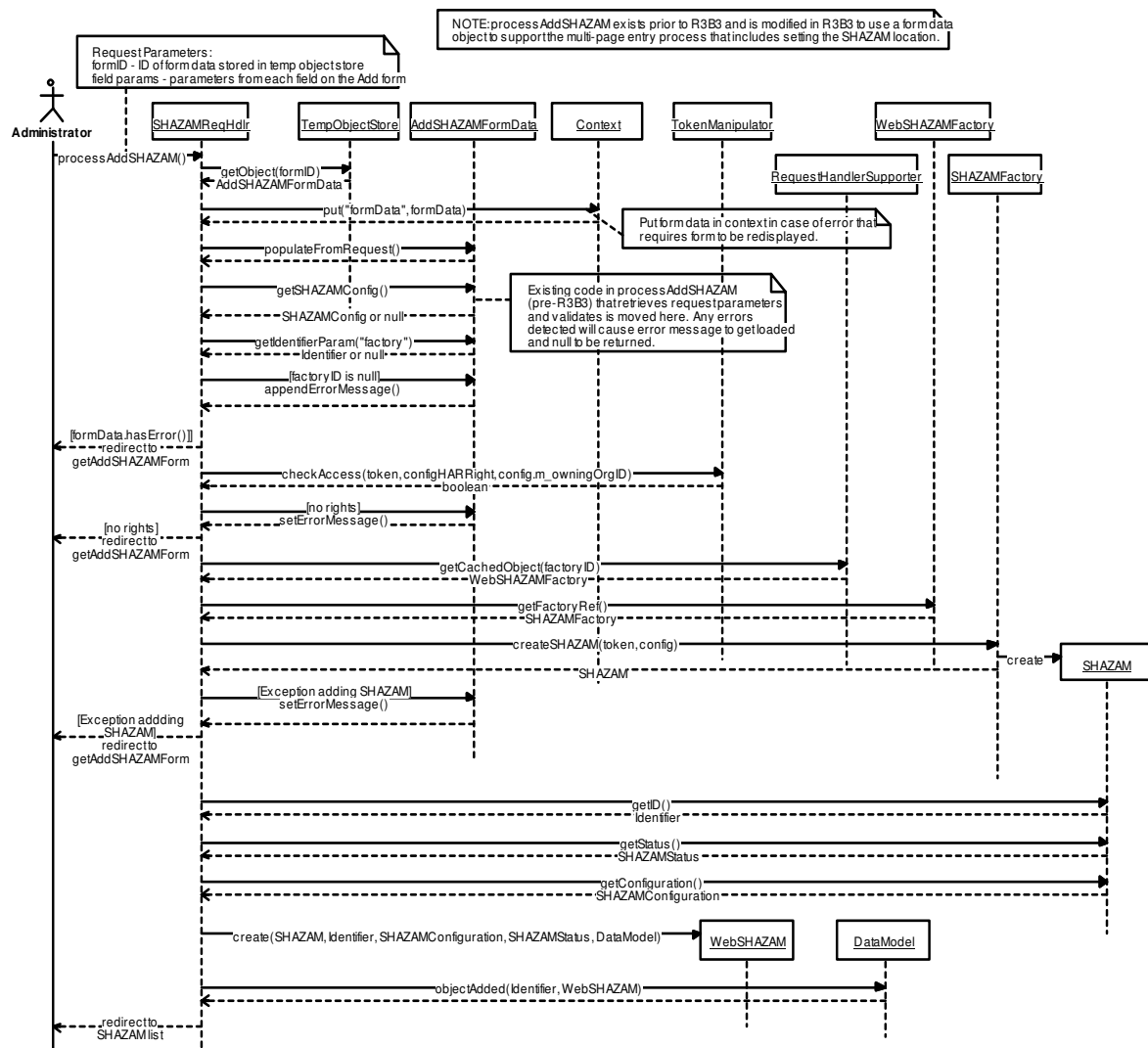


Figure 5-467. SHAZAMReqHdlr:processAddSHAZAM (Sequence Diagram)

5.57 Chartlite.servlet.har

5.57.1 Class diagrams

5.57.1.1 GUIHARServletClasses (Class Diagram)

This diagram shows classes used by the servlet to process requests related to HAR devices.

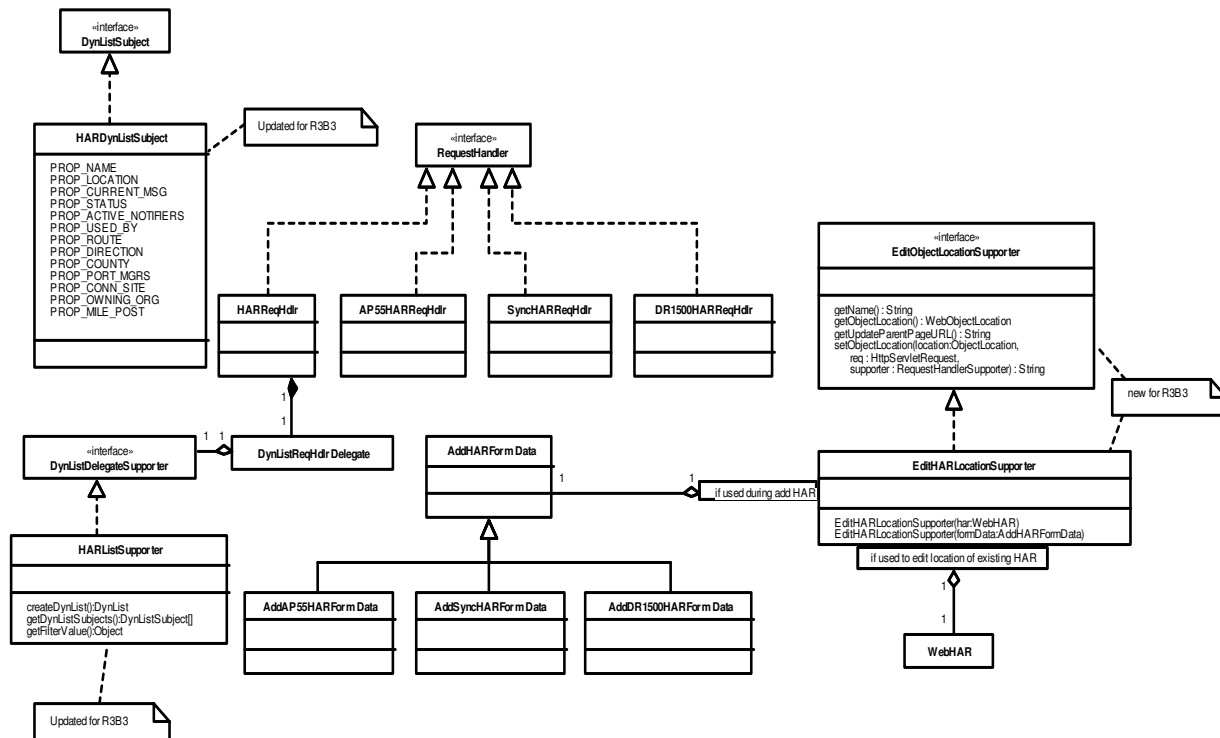


Figure 5-468. GUIHARServletClasses (Class Diagram)

5.57.1.1.1 AddAP55HARFormData (Class)

This class holds data specific to the AP55 HAR when adding an AP55 HAR to the system.

5.57.1.1.2 AddDR1500HARFormData (Class)

This class holds data specific to a DR1500 HAR when adding a DR1500 HAR to the system.

5.57.1.1.3 AddHARFormData (Class)

This class is used to store configuration data for a HAR during an Add operation, as the

operation can require several web pages to complete and the data from each form must be stored between requests.

5.57.1.1.4 AddSyncHARFormData (Class)

This class holds data specific to a Sync HAR when adding a Sync HAR to the system.

5.57.1.1.5 AP55HARReqHdlr (Class)

This class handles requests that are specific to the AP55 HAR model.

5.57.1.1.6 DR1500HARReqHdlr (Class)

This class handles requests that are specific to the DR1500 HAR model.

5.57.1.1.7 DynListDelegateSupporter (Class)

This interface contains functionality to support the DynListReqHdlrDelegate

5.57.1.1.8 DynListReqHdlrDelegate (Class)

This class helps request handlers support dynamic lists. Requests to view, sort, or filter dynamic lists can be passed from a request handler to this class, provided the URL used for the requests contain parameters required by this class, such as the id of the list, the property name, and/or the filter value.

5.57.1.1.9 DynListSubject (Class)

This interface is implemented by classes that wish to be capable of being displayed in a dynamic list.

5.57.1.1.10 EditHARLocationSupporter (Class)

This class implements the EditObjectLocationSupporter interface for HARs. It can be constructed with an AddHARFormData object so it can be used during the Add HAR operation, in which case all location changes are stored in the AddHARFormData object. It can also be constructed using a WebHAR, for use when editing the location of an existing HAR. When this is done, the new location gets set into the actual HAR object (via a CORBA call).

5.57.1.1.11 EditObjectLocationSupporter (Class)

This interface provides functionality allowing the location data to be edited. (For example, the target of the edited location may be an existing object, or it may be a form data object for creating a new object).

5.57.1.1.12 HARDynListSubject (Class)

This class is a dyn list subject that holds a WebHAR. It also defines the property names for each column that will be shown in the list. The following columns are added in R3B3:

route, direction, port managers, connection site, owning org, and mile post.

5.57.1.1.13HARListSupporter (Class)

This class is a dyn list delegate supporter for the HAR list. It provides methods to create a dynamic list, get the subjects included in the list, and to get the value for a filter.

5.57.1.1.14HARReqHdlr (Class)

This class handles requests that are valid for any HAR model.

5.57.1.1.15RequestHandler (Class)

This interface specifies methods that are to be implemented by classes that are used to process requests.

5.57.1.1.16SyncHARReqHdlr (Class)

This class handles requests that are specific to the Sync HAR model.

5.57.1.1.17WebHAR (Class)

This class is a GUI wrapper for a CORBA HAR object.

5.57.2 Sequence Diagrams

5.57.2.1 EditHARLocationSupporter:setObjectLocation (Sequence Diagram)

This diagram shows the processing that is performed when the EditHARLocationSupporter's setObjectLocation method is called. This method is called by the generic location editing request handler when the user submits the edit location form. If the supporter was constructed using a form data object (which is the case if an Add HAR operation is in progress), the location data stored inside the form data is updated and the method returns. If the supporter was constructed using a WebHAR, its CORBA object reference is called to set the location data in the CORBA HAR object. The location data in the WebHAR is updated and the method returns.

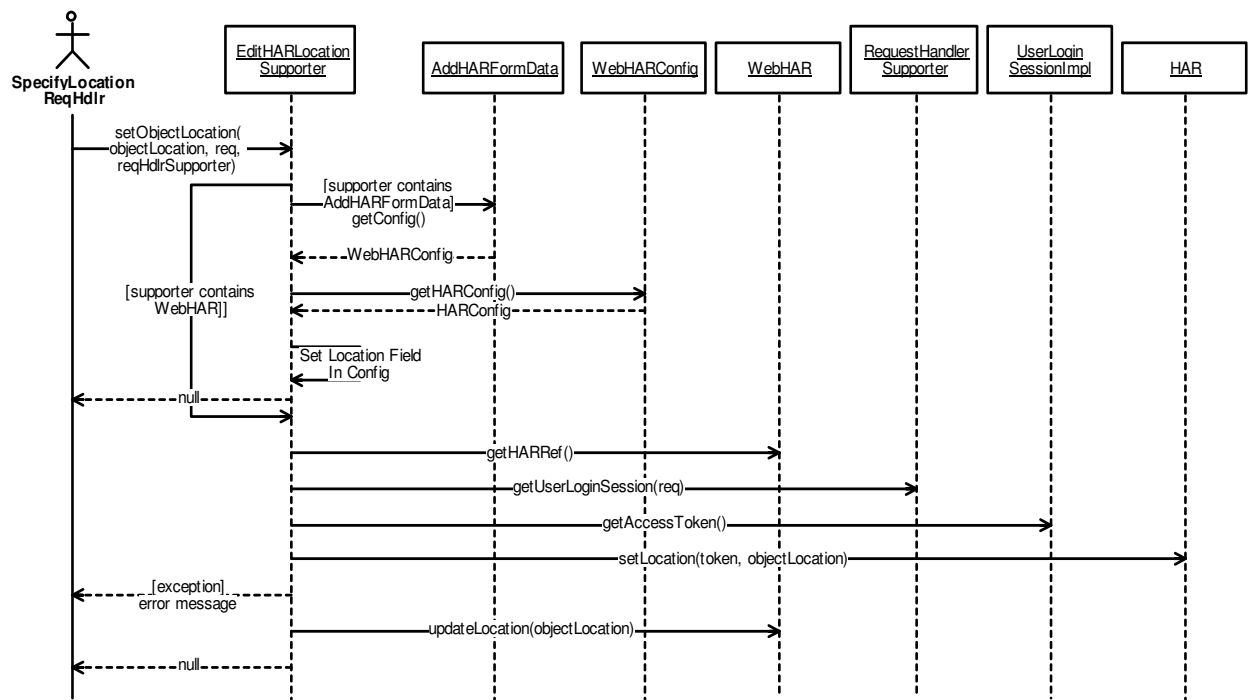


Figure 5-469. EditHARLocationSupporter:setObjectLocation (Sequence Diagram)

5.57.2.2 HARListSupporter:createDynList (Sequence Diagram)

This diagram shows the processing that is performed when the HARListSupporter is called to create a HAR dynamic list. A DefaultDynListCol object is created for each column and stored in an ArrayList. A derivation of DynListComparator is constructed and added to each column, and a BaseDynListFilter derived object is also added to the column if the column supports filtering. The list of columns is used to construct a DefaultDynList. A ConstituentHARFilter is then added to the dyn list as a global filter to initially hide constituents of synchronized HARs. The filterType parameter (if present) is used to make other filter related configurations to the dyn list. If the filterType parameter is set to "OpCenterFolders", an OpCenterFolderFilter is constructed, configured for the user's op center, and added to the dyn list as a global filter which will show only HARs that exist in folders that are tagged for use with the user's op center. Other filterType values are used to set the status filter, and in the case of the "OnlineWithMsg" filter type, the message filter.

This processing is updated in R3B3 to add support for the following columns: Route, Direction, County, Port Managers, Connection Site, Owning Org, Mile Post.

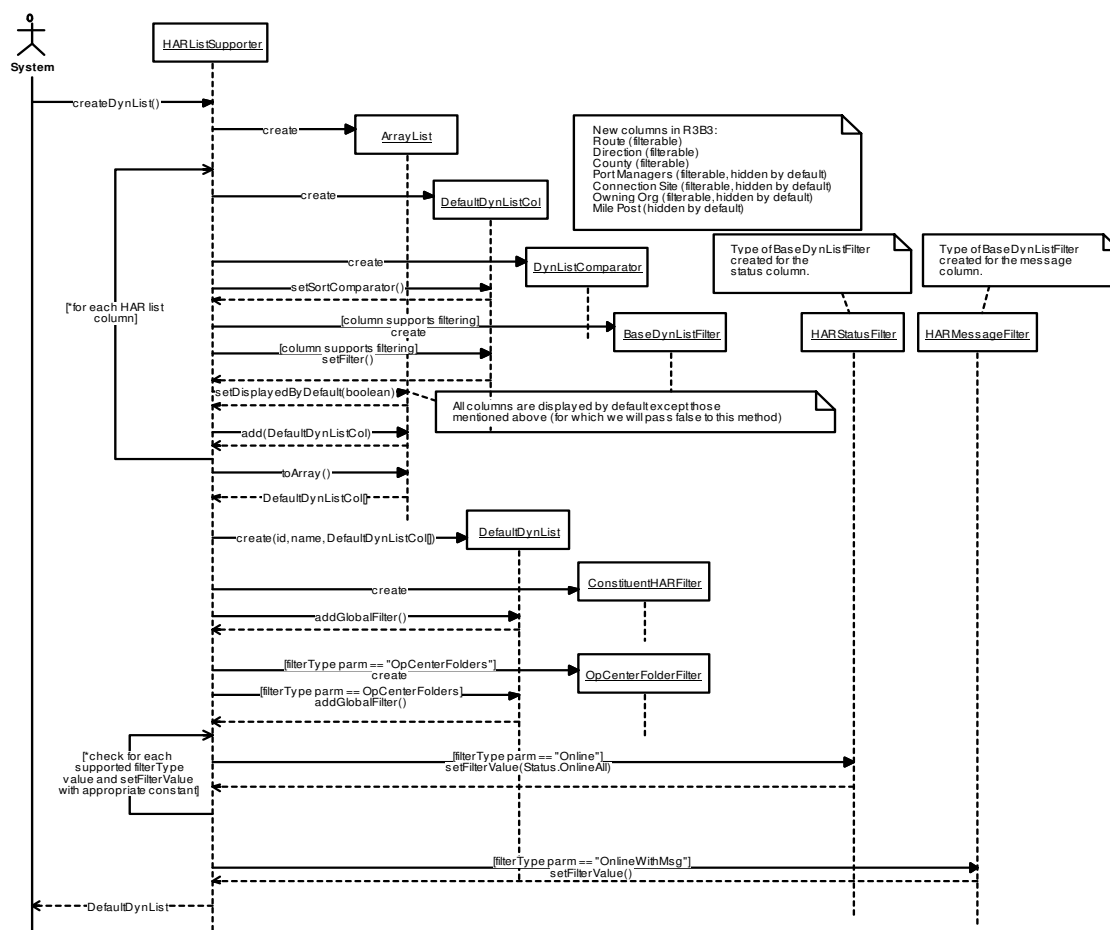


Figure 5-470. HARListSupporter:createDynList (Sequence Diagram)

5.57.2.3 HARReqHdlr:getEditHARLocationForm (Sequence Diagram)

This diagram shows the processing that is performed when the user chooses to edit the location fields for a HAR. This can be done as part of the Add HAR process (any model), or can be done on an existing HAR from its details page. If done during the AddHAR process, a tmpObjID parameter will be present and is used to retrieve the AddHARFormData object from the temp object store. The parseFormData method is called to store any data the user had entered in fields of the Add HAR form prior to choosing to set the location fields. An EditHARLocationSupporter object is then created, passing the AddHARFormData to the constructor.

If a harID is passed in the request, this indicates the operation is being performed "stand alone" from the HAR details page. In this case, the user rights must be checked, and the WebHAR is retrieved from the object cache. An EditHARLocationSupporter is then constructed from the WebHAR object.

After the EditHARLocationSupporter is created (either with AddHARFormData or a WebHAR), it is placed in the temp object store so it can be accessed by the generic location editing request handler. The request is then redirected to the request that shows the generic edit location form.

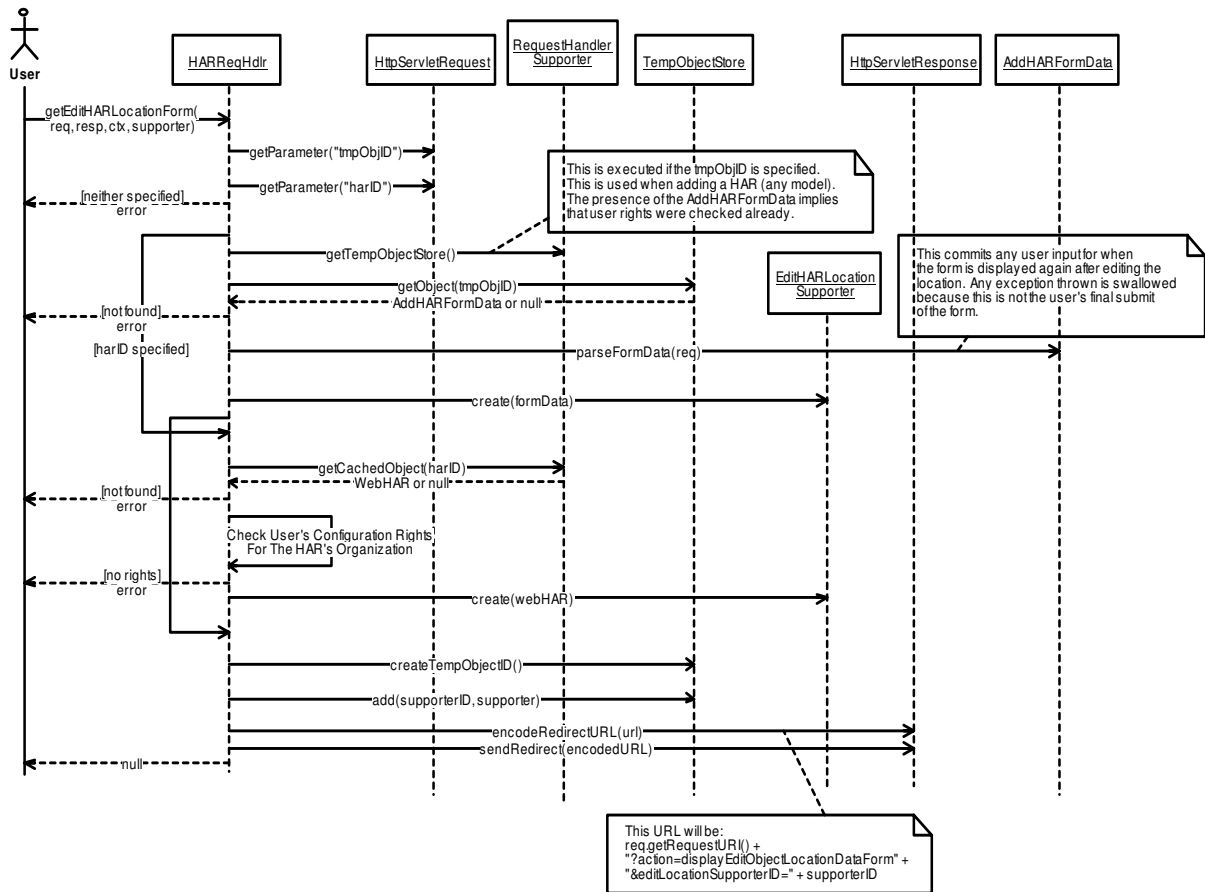


Figure 5-471. HARReqHdlr::getEditHARLocationForm (Sequence Diagram)

5.58 Charlite.Flex.shared.components-flex

5.58.1 Class Diagrams

5.58.1.1 GUIFlexComponentsClasses (Class Diagram)

This diagram shows common Flex components.

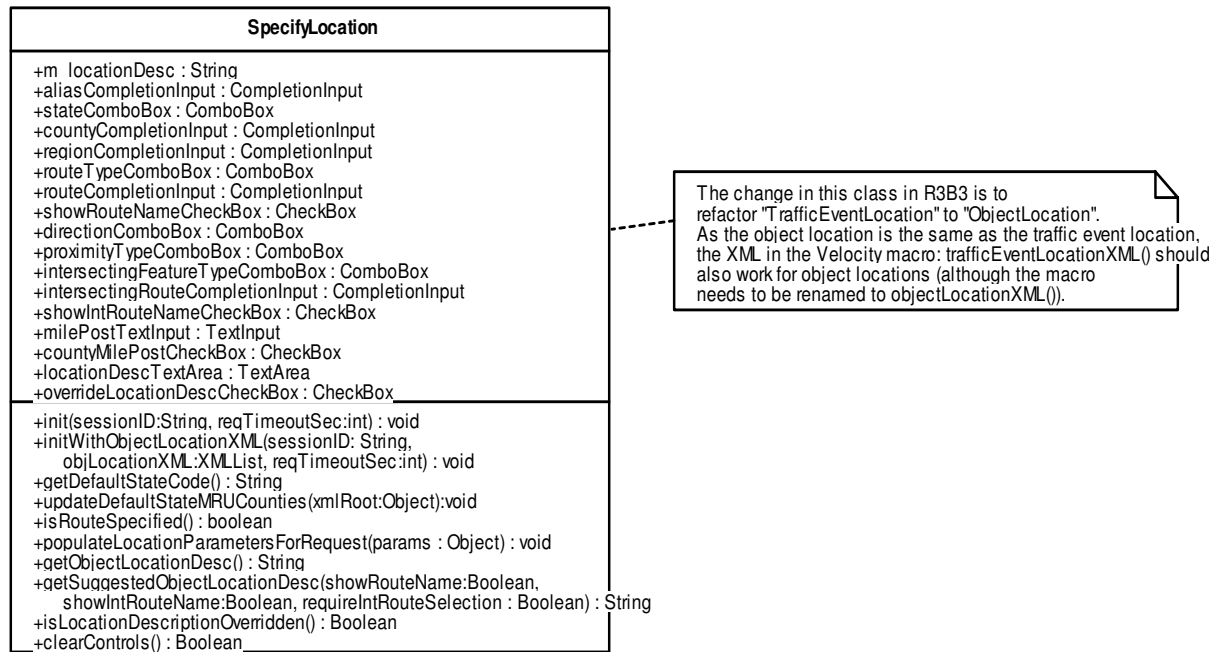


Figure 5-472. GUIFlexComponentsClasses (Class Diagram)

5.58.1.1.1 SpecifyLocation (Class)

This is a Flex control that allows the location fields to be specified.

5.59 Chartlite.Flex.editlocation

5.59.1 Class Diagrams

5.59.1.1 GUIFlexEditLocationClasses (Class Diagram)

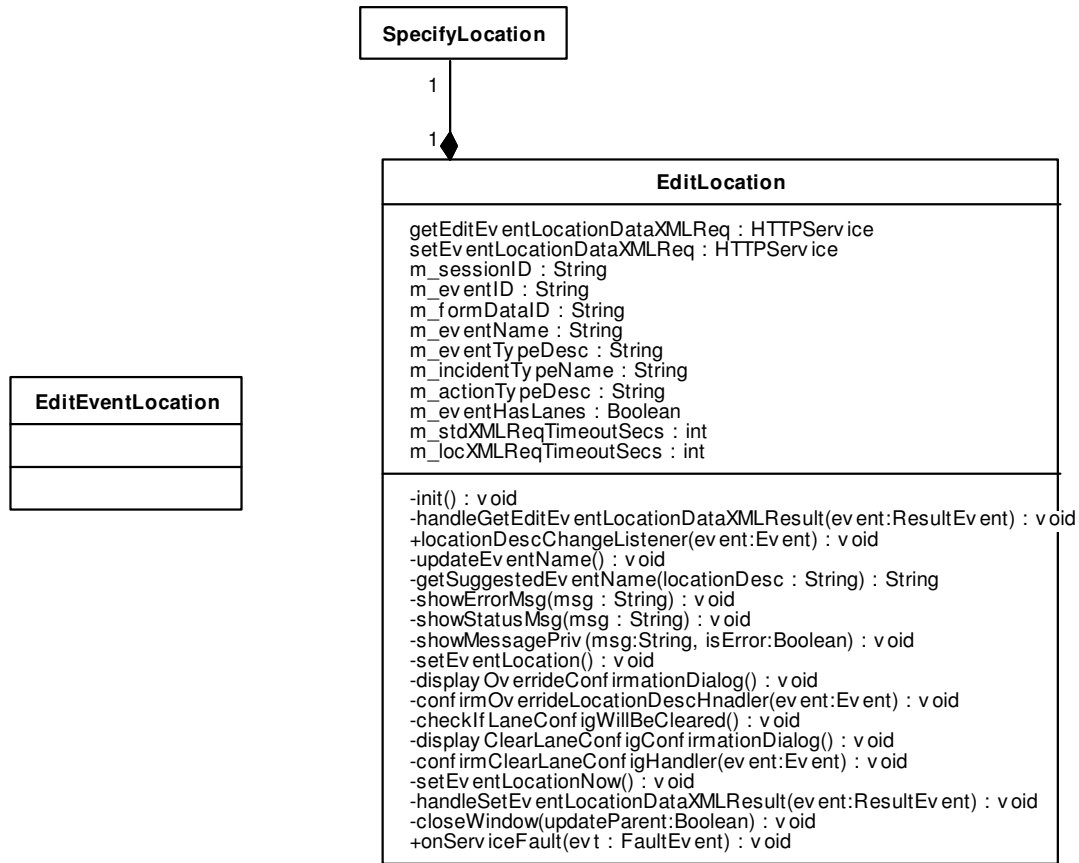


Figure 5-473. GUIFlexEditLocationClasses (Class Diagram)

5.59.1.1.1 EditEventLocation (Class)

5.59.1.1.2 EditLocation (Class)

5.59.1.1.3 SpecifyLocation (Class)

This is a Flex control that allows the location fields to be specified.

5.60 Chartlite.util

5.60.1 Classes

5.60.1.1 chartlite.util_classes (Class Diagram)

This diagram shows utility classes used in the CHART GUI servlet.

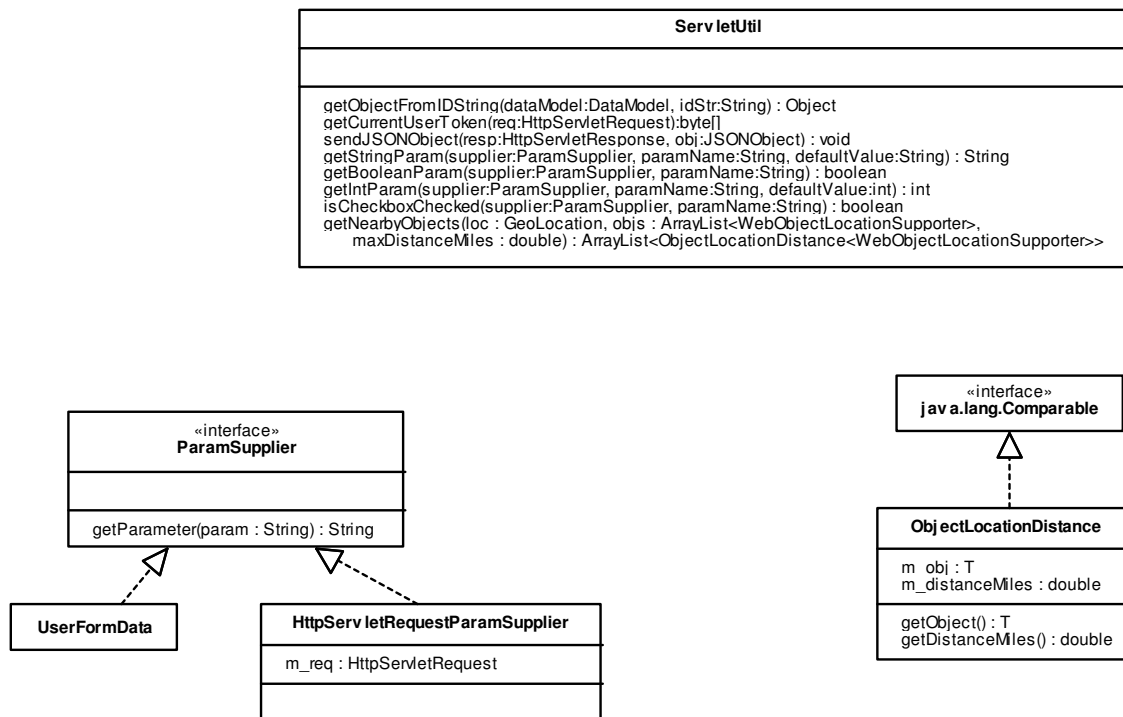


Figure 5-474. chartlite.util_classes (Class Diagram)

5.60.1.1.1 HttpServletRequestParamSupplier (Class)

This class implements the ParamSupplier interface to provide parameters from an HttpServletRequest object.

5.60.1.1.2 java.lang.Comparable (Class)

This interface allows two objects to be compared for the purposes of sorting.

5.60.1.1.3 ObjectLocationDistance (Class)

This class stores an object and a calculated distance to another point. It is used to avoid re-calculating the distance multiple times.

5.60.1.1.4 ParamSupplier (Class)

This interface allows parameter values to be retrieved. It was added to handle parameters supplied by a HttpServletRequest and form data using common code.

5.60.1.1.5 ServletUtil (Class)

This class provides static utility methods useful to request handlers in the servlet.

5.60.1.1.6 UserFormData (Class)

This class is used to store form data between requests while a user is editing a complex form, and provides convenience methods for parsing the values from the request.

5.60.2 Sequence Diagrams

5.60.2.1 ServletUtil:getNearbyObjects (Sequence Diagram)

This diagram shows the processing to find the objects near a given location, sorted by distance. When `getNearbyObjects()` is called, it creates a list for storing object / distance pairs. Each object from the input list is called to get its `GeoLocation`, which can be null if the geo location is not specified for an object. If the geo location is not null, the distance to the geo location is calculated. Then if the distance is less than the specified maximum distance, or if the maximum distance is unspecified, an `ObjectLocationDistance` object is created to store the object and the distance, and this is added to the list. The list is sorted by distance and returned to the caller.

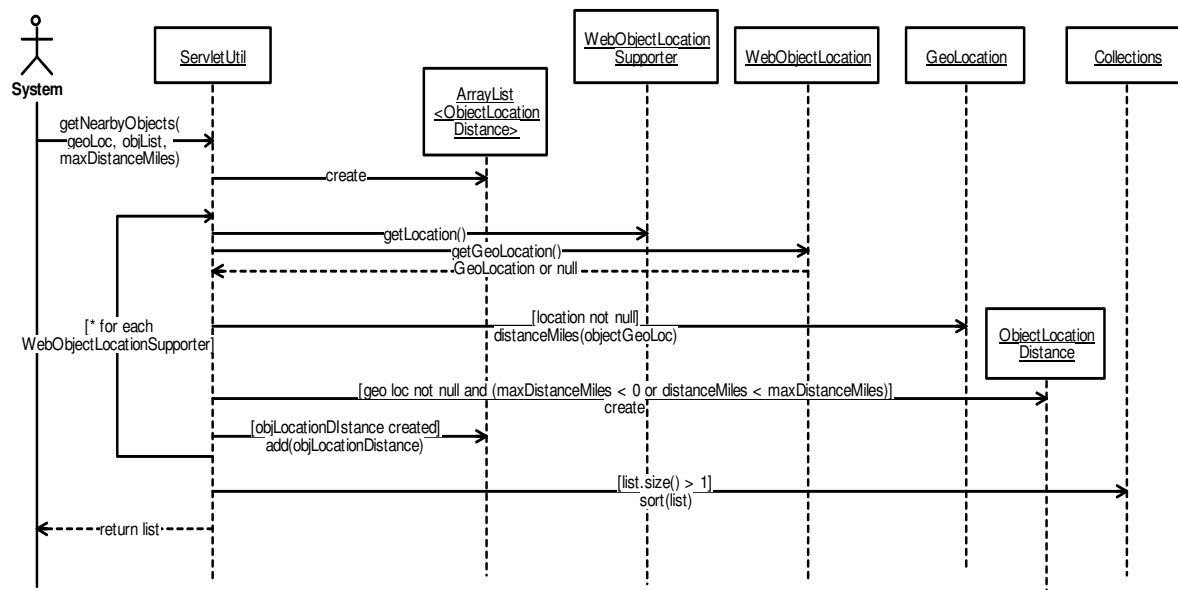


Figure 5-475. ServletUtil:getNearbyObjects (Sequence Diagram)

5.61 Chartlite.util.dynlist

5.61.1 Class Diagrams

5.61.1.1 DynamicListClasses (Class Diagram)

This diagram shows interfaces and classes that provide generic support for dynamic lists. A dynamic list is a list of objects that has one or more columns and can be sorted and filtered.

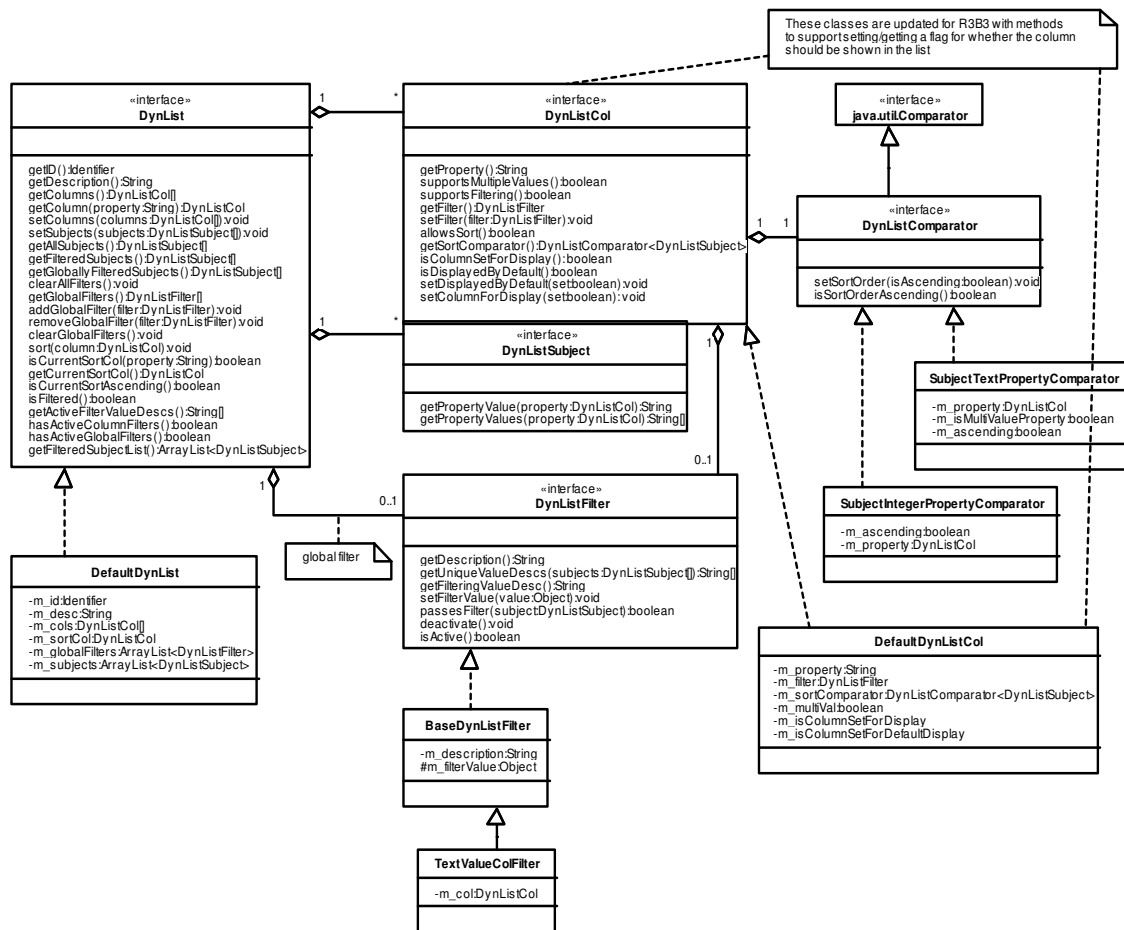


Figure 5-476. DynamicListClasses (Class Diagram)

5.61.1.1.1 BaseDynListFilter (Class)

This abstract class provides a base implementation of the DynListFilter interface.

5.61.1.1.2 DefaultDynList (Class)

This class provides a default implementation of the DynList interface. It supports a collection of columns, a collection of global filters, and a collection of subjects. Filters in this list are treated additively - that is, a subject must pass all filters to be displayed.

5.61.1.1.3 DefaultDynListCol (Class)

This class provides a default implementation of the DynListCol interface. This column is constructed with a string property name for which subjects are expected to provide a value. By default, this column uses a SubjectTextPropertyComparator, which means a string comparison of the property values provided by the subjects for this column is used. You may optionally set a different comparator. Multiple values for this column (from a single subject) are supported.

5.61.1.1.4 DynList (Class)

This interface is implemented by classes that wish to provide dynamic list capabilities. A dynamic list is a list of items that has one or more columns that can optionally be sorted, and the list can be filtered by column values or by global filters.

5.61.1.1.5 DynListCol (Class)

This interface is implemented by classes that are to be used as a column in a dynamic list.

5.61.1.1.6 DynListComparator (Class)

This interface is implemented by classes that are used to sort dynamic lists.

5.61.1.1.7 DynListFilter (Class)

This interface is implemented by classes that are used to filter dynamic lists.

5.61.1.1.8 DynListSubject (Class)

This interface is implemented by classes that wish to be capable of being displayed in a dynamic list.

5.61.1.1.9 java.util.Comparator (Class)

This interface is implemented by classes that can be sorted.

5.61.1.1.10 SubjectIntegerPropertyComparator (Class)

This class is a dyn list comparator that can be used to sort columns that contain integer

values.

5.61.1.1.11 SubjectTextPropertyComparator (Class)

This class provides an implementation of the DynListComparator interface which compares subjects based on the values they supply for the property supplied to this class during construction. A case insensitive text comparison is done on the values, and multiple value columns are supported.

5.61.1.1.12 TextValueColFilter (Class)

This class is a DynListFilter that filters subjects of a dynamic list based on the text value of a column's property.

6 Mapping To Requirements

The following table shows how the requirements in the CHART R3B3 Requirements document map to design elements contained in this design.

Those requirements that are included for reference but are not to be implemented in R3B3 are shaded in grey; those that are not shaded in grey are requirements to be implemented in R3B3. Grey requirements may be marked “[FUTURE]”, meaning they are existing baseline requirements which are included for reference but are to be implemented in a future release. These may be existing or new requirements. If new, they are marked as “[FUTURE, NEW]”. Grey requirements may be marked as “[NEW BUT ALREADY IMPLEMENTED]” meaning they have already been implemented in the system and are being added to the baseline to reflect the as-built system. Requirements revised from the baseline are marked with an asterisk: ‘*’.

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1	ADMINISTER SYSTEMS AND EQUIPMENT	HEADER	N/A	N/A
1.1	ADMINISTER CHART ORGANIZATIONS, LOCATIONS, AND USERS	HEADER	N/A	N/A
1.1.1	MAINTAIN CHART ORGANIZATIONS AND GEOGRAPHIC AREAS OF RESPONSIBILITY. The system shall allow the user to separately specify identify organizations, types of locations, and geographic areas of responsibility, and to associate them to each other.	HEADER	N/A	N/A
1.1.1.2	MAINTAIN GEOGRAPHIC AREAS OF RESPONSIBILITY [FUTURE]	HEADER	N/A	N/A
1.1.1.2.3	The system shall allow the user to define map-based areas of responsibility. Suggestions/examples to be validated: regions, county boundaries, city boundaries. [FUTURE]	FUTURE	N/A	N/A
1.1.1.2.3.1	The system shall allow a suitably privileged user to define named geographical areas to serve as components (building blocks) for areas of responsibility.	Geographical	Configure Geographical Settings	GeoAreaModulePkg CD, GeoAreaModulePkg:initialize SD, GeoAreaModuleImpl:shutdown SD
1.1.1.2.3.1.1	The system shall allow a suitably privileged user to add a geographical area.	Geographical	Add Geographical Area	displayAddEditGeoAreaForm, submitAddEditGeoAreaForm. GeoAreaFactoryImpl:addGeoArea SD

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.1.1.2.3.1.1.1	A geographical area shall include a name.	Geographical	Add Geographical Area	displayAddEditGeoAreaForm, submitAddEditGeoAreaForm, GeoAreaModulePkg CD
1.1.1.2.3.1.1.2	A geographical area shall be defined as a polygon, with vertices specified in geographical (latitude/longitude) coordinates.	Geographical	Add Geographical Area	displayAddEditGeoAreaForm, submitAddEditGeoAreaForm, GeoAreaModulePkg CD
1.1.1.2.3.1.1.2.1	The system shall allow a user to add a point to the polygon by specifying geographical coordinates.	Geographical	Specify Polygon	displayAddEditGeoAreaForm, submitAddEditGeoAreaForm, GeoAreaModuleImpl:addGeoArea SD
1.1.1.2.3.1.1.2.2	The system shall allow a user to modify the coordinates of a point in the polygon.	Geographical	Specify Polygon	displayAddEditGeoAreaForm, submitAddEditGeoAreaForm, GeoAreaModuleImpl:updateGeoArea SD
1.1.1.2.3.1.1.2.3	The system shall allow a user to remove a point from the polygon.	Geographical	Specify Polygon	displayAddEditGeoAreaForm, submitAddEditGeoAreaForm, GeoAreaModuleImpl:updateGeoArea SD
1.1.1.2.3.1.1.2.4	The system shall allow a user to define the polygon by importing a file in Keyhole Markup Language (KML) format.	Geographical	Specify Polygon, Import KML File	importKMLFileJSON
1.1.1.2.3.1.2	The system shall allow a suitably privileged user to modify a geographical area, following the same rules as when adding a geographical area.	Geographical	Edit Geographical Area	displayAddEditGeoAreaForm, submitAddEditGeoAreaForm, GeoAreaModuleImpl:updateGeoArea SD
1.1.1.2.3.1.3	The system shall allow a suitably privileged user to delete a geographical area.	Geographical	Delete Geographical Area	removeGeoArea, GeoAreaModuleImpl:removeGeoArea SD
1.1.1.2.3.1.3.1	The system shall prompt the user for confirmation before removing a geographical area.	Geographical	Delete Geographical Area	removeGeoArea
1.1.1.2.3.1.3.2	The system shall not allow a geographical area to be removed if it is referenced in the system.	Geographical	Delete Geographical Area	removeGeoArea, GeoAreaModuleImpl:removeGeoArea SD
1.1.1.2.3.2	The system shall validate any geographical coordinates entered by a user as freeform text against system-wide bounds.	Geographical	Specify Polygon	submitAddEditGeoAreaForm
1.1.1.2.3.2.1	The system shall allow a suitably privileged user to specify the system-wide bounds for acceptable user-entered coordinates.	Geographical	Configure Geographical Settings	
1.1.1.2.3.2.1.1	The system shall allow a suitably privileged user to define the minimum acceptable latitude (southern boundary).	Geographical	Configure Geographical Settings	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.1.1.2.3.2.1.2	The system shall allow a suitably privileged user to define the maximum acceptable latitude (northern boundary).	Geographical	Configure Geographical Settings	
1.1.1.2.3.2.1.3	The system shall allow a suitably privileged user to define the minimum acceptable longitude (western boundary).	Geographical	Configure Geographical Settings	
1.1.1.2.3.2.1.4	The system shall allow a suitably privileged user to define the maximum acceptable longitude (eastern boundary).	Geographical	Configure Geographical Settings	
1.4	MAINTAIN CHART CONTROL	HEADER	N/A	
1.4.6	MANAGE ALERTS	HEADER	N/A	
1.4.6.1	A suitably privileged user shall be able to specify an “ignore” system property for an alert type, which indicates that no user will ever see alerts of this type at any time. This provides a “back-out” capability to completely ignore alerts of this type.	EXISTING (extending for new alerts)	Configure R3B3 Alert Types	data.alerts.classes CD getAlertTimeoutsAndPoliciesForm, setAlertTimeoutsAndPolicies
1.4.6.2	A suitably privileged user shall be able to define the default and maximum values for timeouts related to alert processing.	EXISTING (extending for new alerts)	Configure R3B3 Alert Types	data.alerts.classes CD getAlertTimeoutsAndPoliciesForm, setAlertTimeoutsAndPolicies
1.4.6.2.1	A suitably privileged user shall be able to specify the default Accept timeout for each alert type.	EXISTING (extending for new alerts)	Configure R3B3 Alert Types	data.alerts.classes CD getAlertTimeoutsAndPoliciesForm, setAlertTimeoutsAndPolicies
1.4.6.2.2	A suitably privileged user shall be able to specify the maximum Accept timeout for each alert type.	EXISTING (extending for new alerts)	Configure R3B3 Alert Types	data.alerts.classes CD getAlertTimeoutsAndPoliciesForm, setAlertTimeoutsAndPolicies
1.4.6.2.3	A suitably privileged user shall be able to specify the default alert Delay timeout for each alert type.	EXISTING (extending for new alerts)	Configure R3B3 Alert Types	data.alerts.classes CD getAlertTimeoutsAndPoliciesForm, setAlertTimeoutsAndPolicies
1.4.6.2.4	A suitably privileged user shall be able to specify the maximum Delay timeout for each alert type.	EXISTING (extending for new alerts)	Configure R3B3 Alert Types	data.alerts.classes CD getAlertTimeoutsAndPoliciesForm, setAlertTimeoutsAndPolicies

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.4.6.2.5	A suitably privileged user shall be able to specify the escalation timeout for each alert type.	EXISTING (extending for new alerts)	Configure R3B3 Alert Types	data.alerts.classes CD getAlertTimeoutsAndPoliciesForm, setAlertTimeoutsAndPolicies
1.4.6.2.6	A suitably privileged user shall be able to specify the archive timeout for Closed alerts, after which a Closed alert is removed from the active CHART system and is able to be sent to an alert archive.	EXISTING	N/A	
1.4.6.2.7	A suitably privileged user shall be able to disable escalation of alerts of a particular alert type for which escalation is currently not desired.	EXISTING (extending for new alerts)	Configure R3B3 Alert Types	data.alerts.classes CD
1.5	INSTALL AND MAINTAIN DEVICES	HEADER	N/A	
1.5.2	PUT EQUIPMENT/ DEVICES ON-LINE.	HEADER	N/A	
1.5.2.1	The system shall allow the user with appropriate rights to select (or modify) the equipment device parameters.	EXISTING	N/A	
1.5.2.1.4	The system shall support configuration parameters for DMS devices.	EXISTING	View DMS Details	GUIDMSDataClasses DMSControlModule: DMSControlClassDiagram DMSControlModule:GetConfiguration
1.5.2.1.4.9	The system shall support setting a responsible Center for a DMS which is to receive the Device Failure Alert when the device goes into communication failure.	DMS	Configure DMS	parseBasicConfigSettings DMSControlClassDiagram DMSControlModule:handleOpstatus DMSControlModule:SetConfiguration DMSControlModule:GetConfiguration
1.5.2.1.4.10	The system shall support setting a notification group to receive DMS communication failure notification messages.	DMS	Configure DMS	parseBasicConfigSettings DMSControlClassDiagram DMSControlModule:handleOpstatus DMSControlModule:SetConfiguration DMSControlModule:GetConfiguration
1.5.2.1.4.11	The system shall support setting a notification group to receive DMS hardware failure notification messages.	DMS	Configure DMS	parseBasicConfigSettings DMSControlClassDiagram DMSControlModule:handleOpstatus DMSControlModule:SetConfiguration DMSControlModule:GetConfiguration

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.2.1.4.12	The system shall support setting the device port for communication with the device.	DMS	Configure DMS, Specify TCPIP Device Connection	
1.5.2.1.4.12.4	The system shall support setting the IP Address and TCP port number for a DMS that supports TCP/IP protocol.	DMS	Configure DMS, Specify TCPIP Device Connection, View TCP/IP Port Configuration	DMSControlClassDiagram DeviceUtilityPkg:PortLocatorClasses PortLocator:getPort PortLocator:getconnectedPort2 PortLocator:ReleasePort2
1.5.2.1.4.13	The system shall support setting a default font number for NTCIP DMS.	DMS	Configure NTCIP DMS, View NTCIP Font Settings	parseBasicConfigSettings DMSControlModuleClassDiagram DMSControlModule:GetConfiguration DMSControlModule:SetConfiguration
1.5.2.1.4.14	The system shall support setting default line spacing for NTCIP DMS.	DMS	Configure NTCIP DMS, View NTCIP Font Settings	parseBasicConfigSettings DMSControlModuleClassDiagram DMSControlModule:GetConfiguration DMSControlModule:SetConfiguration
1.5.2.1.4.15	The system shall support setting the DMS location as specified in 1.5.2.4	DMS	Configure DMS, Set Device Location	DMSReqHdlr:getEditDMSLocationForm, EditDMSLocationSupporter:setObjectLocation DMSControlModuleClassDiagram
1.5.2.1.4.16	The system shall support setting a travel time display schedule for a DMS.	DMS	Set DMS Travel Time Display Schedule, View Travel Time Message Schedule	setDMSTravelTimeDisplaySchedule, Chart2DMSImpl:setTravelTimeSchedule
1.5.2.1.4.17	The system shall support associating travel routes to a DMS.	DMS	Add Travel Route to DMS, Remove Travel Route from DMS	viewEditDMSTravelRoutesForm, setDMSTravelRoutes, Chart2DMSImpl:setRelatedRoutes
1.5.2.1.4.18	The system shall support setting the arbitration queue level at which a travel time message should be added to the arbitration queue for a DMS.	DMS	Specify DMS Traveler Information Message Settings	parseBasicConfigSettings, Chart2DMSImpl:setQueueLevels
1.5.2.1.4.18.1	The system shall consider any traveler information message that contains at least one travel time field but no toll rate fields a travel time message.	DMS	Activate Traveler Information Message	DMSTravInfoMsgHandler:checkMessage

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.2.1.4.18.2	The arbitration queue level at which travel time messages are added shall default to the Travel Time level for a DMS.	DMS	Specify DMS Traveler Information Message Settings	parseBasicConfigSettings
1.5.2.1.4.19	The system shall support setting the arbitration queue level at which a toll rate message should be added to the arbitration queue for a DMS.	DMS	Specify DMS Traveler Information Message Settings	parseBasicConfigSettings, Chart2DMSImpl:setQueueLevels
1.5.2.1.4.19.1	The system shall consider any traveler information message that contains at least one toll rate field as a toll rate message (regardless of other fields the message may contain such as travel time fields).	DMS	Activate Traveler Information Message	DMSTravInfoMsgHandler:checkMessage
1.5.2.1.4.19.2	The arbitration queue level at which toll rate messages are added shall default to the Toll Rate level for a DMS.	DMS	Specify DMS Traveler Information Message Settings	parseBasicConfigSettings
1.5.2.1.5	The system shall allow a suitably privileged user to add a new device which communicates via an implemented protocol over an implemented communications medium.	EXISTING	N/A	DeviceUtility:PortLocatorClasses
1.5.2.1.5.6	The system shall support DMS communication via a TCP/IP communications medium.	DMS	Communicate With DMS	DeviceUtility:PortLocatorClasses
1.5.2.1.5.7	The system shall support TSS communication via a TCP/IP communications medium.	TSS	Communicate With TSS	
1.5.2.1.7	The system shall provide an API to implement DMS functionality specified in the NTCIP standard NEMA TS3.6.	DMS	Set NTCIP DMS Message	
1.5.2.1.7.1	The system shall automatically set the default Font number before displaying a message for a NTCIP DMS.	DMS	Set NTCIP DMS Message	NTCIPProtocolHandler:SetMessage
1.5.2.1.7.2	The system shall automatically set the default line spacing before displaying a message for an NTCIP DMS.	DMS	Set NTCIP DMS Message	NTCIPProtocolHandler:SetMessage
1.5.2.1.8	Phone numbers associated with a HAR will be viewable through the Properties option for that HAR in the Navigator. The system shall support configuration parameters for HAR devices. *	EXISTING	View HAR Details	GUIVideoDataClasses CD
1.5.2.1.8.4	The system shall support setting device location for a HAR as specified in 1.5.2.4.	HAR	Configure HAR, Set Device Location	GUIHARClasses, GUIHARServletClasses, getEditHARLocationForm, EditHARLocationSupporter.setObjectLocation HARControlModule CD

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.2.1.9	The GUI shall provide the capability to show the id of each device. The system shall support configuration parameters for SHAZAM devices. *	EXISTING	View SHAZAM Details	GUISHAZAMClasses
1.5.2.1.9.10	The system shall support setting the location of a SHAZAM as specified in 1.5.2.4.	SHAZAM	Configure SHAZAM, Set Device Location	GUISHAZAMClasses, GUISHAZAMServletClasses, getSHAZAMEditLocationForm, EditSHAZAMLocationSupporter:setObjectLocation SHAZAMControlModule CD
1.5.2.1.17	The system shall support configuration parameters for TSS (Traffic Sensor System) devices (detectors).	EXISTING	N/A	
1.5.2.1.17.7	The system shall support setting TSS location as specified in 1.5.2.4.	TSS	Configure Detector, Set Device Location	TSSReqHdlr:getEditTSSLocationForm, EditTSSLocationSupporter:setObjectLocation TSSControlModuleCD
1.5.2.1.17.8	The system shall support setting a responsible Center for a TSS which is to receive the Device Failure Alert when the device goes into hardware failure.	EXISTING	N/A	
1.5.2.1.17.9	The system shall support setting IP Address and TCP port number for a TSS that supports TCP/IP protocol.	TSS	Configure Detector, Specify TCPIP Device Connection, View TCPIP Port Configuration	setTSSConfigCommSettings TSSManagementModulePackage CD DeviceUtility:PortLcoatorClasses PolledTSSImpl:setConfiguration
1.5.2.1.18	1.5.2.1.18 The system shall support setting configuration parameters for Cameras. [NEW BUT ALREADY IMPLEMENTED]	EXISTING	View Camera Details	GUIVideoDataClasses CD
1.5.2.1.18.1	The system shall support setting the device location of a camera as in 1.5.2.4.	Video	Configure Camera, Set Device Location	EditCameraLocationSupporter:setObjectLocation, VideoSourceConfigReqHdlr:getEditCameraLocationForm Common2 CD VideoHighLevel-VideoSource
1.5.2.1.20	The system shall not allow the user to change the configuration parameters for an External Device.	Device	Configure DMS, Configure Detector	GUIDMSDataClasses GUITSSDataClasses ExternalDMS:SetExternalConfiguration

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.2.4	The system shall allow the user with appropriate rights to select specify the equipment location by lat/long when supported by the equipment type. *	HEADER	View Device Location Details	chartlite.servlet.location_classes, SpecifyLocationReqHdlr:displayEditObjectLocationDataForm, SpecifyLocationReqHdlr:getEditObjectLocationDataXML, SpecifyLocationReqHdlr:setObjectLocationDataXML
1.5.2.4.1	The system shall support viewing and setting the State where a device is located.	Device	Set Device Location	chartlite.servlet.location_classes Common2 CD
1.5.2.4.2	The system shall support viewing and setting the County where a device is located.	Device	Set Device Location	chartlite.servlet.location_classes Common2 CD
1.5.2.4.3	The system shall support viewing and setting the Route Type where a device is located.	Device	Set Device Location	chartlite.servlet.location_classes Common2 CD
1.5.2.4.4	The system shall support viewing and setting the Route where a device is located.	Device	Set Device Location	chartlite.servlet.location_classes Common2 CD
1.5.2.4.4.1	The system shall support an indicator stating whether the route number or route name is to be used when displaying the route where the device is located.	Device	Set Device Location	chartlite.servlet.location_classes Common2 CD
1.5.2.4.5	The system shall support viewing and setting the Direction of the Route where a device is located.	Device	Set Device Location	chartlite.servlet.location_classes Common2 CD
1.5.2.4.5.1	The system shall support bi-directional directions. (Examples: North/South, East/West, Inner Loop/Outer Loop)	Device	Set Device Location	chartlite.servlet.location_classes Common2 CD
1.5.2.4.6	The system shall support the following proximity descriptions for describing the location of the device relative to the intersecting feature (if any): AT, PAST, PRIOR, NORTH OF, SOUTH OF, WEST OF, EAST OF.	Device	Set Device Location	chartlite.servlet.location_classes Common2 CD
1.5.2.4.7	The system shall support viewing and setting the intersecting route to specify the location of the device along a route.	Device	Set Device Location	chartlite.servlet.location_classes Common2 CD
1.5.2.4.7.1	The system shall support an indicator stating whether the intersecting route number or name is to be used when displaying the route where the device is located.	Device	Set Device Location	chartlite.servlet.location_classes Common2 CD
1.5.2.4.8	The system shall support viewing and setting the State Milepost of the route where a device is located.	Device	Set Device Location	chartlite.servlet.location_classes Common2 CD

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.2.4.9	The system shall support viewing and setting the Latitude where a device is located.	Device	Set Device Location	chartlite.servlet.location_classes Common2 CD
1.5.2.4.10	The system shall support viewing and setting the Longitude where a device is located.	Device	Set Device Location	chartlite.servlet.location_classes Common2 CD
1.5.2.4.12	The system shall support viewing and setting a text description of a device location.	Device	Set Device Location	chartlite.servlet.location_classes Common2 CD
1.5.2.4.12.1	The system shall require the location description to be specified.	Device	Set Device Location	SpecifyLocationReqHdr:setObjectLocationDataXML
1.5.2.4.12.2	The system shall automatically generate the location description based on the other location fields specified.	Device	Set Device Location	
1.5.2.4.12.3	The system shall allow the user to override the system generated location description.	Device	Set Device Location	
1.5.2.4.12.3.1	The system shall warn the user when overriding the generated location description for a device.	Device	Set Device Location	
1.5.2.4.13	The system shall not allow the user to specify the equipment location if the equipment data is imported from an external system.	Device	Configure DMS, Configure Detector	getEditDMSLocationForm, getEditTSSLocationForm Common2 CD DMSControl CD
1.5.5	VIEW DEVICE LISTS	HEADER	N/A	
1.5.5.1	The system shall allow the user to view the list of DMSs that exist in the system.	EXISTING	N/A	
1.5.5.1.1	The system shall allow the user to view detailed data for each DMS in the list	EXISTING	N/A	
1.5.5.1.1.6	The detailed data displayed for a DMS shall include the County where the DMS is located (if specified).	DMS	View DMS List	chartlite.servlet.dms_dynlist_classes, DMSListSupporter:createDynList
1.5.5.1.1.7	The detailed data displayed for a DMS shall include the Route where the DMS is located (if specified).	DMS	View DMS List	chartlite.servlet.dms_dynlist_classes, DMSListSupporter:createDynList
1.5.5.1.1.8	The detailed data displayed for a DMS shall include the Direction of the route where the DMS is located (if specified).	DMS	View DMS List	chartlite.servlet.dms_dynlist_classes, DMSListSupporter:createDynList
1.5.5.1.1.9	The detailed data displayed for a DMS shall include the Port Managers assigned to a DMS.	DMS	View DMS List	chartlite.servlet.dms_dynlist_classes, DMSListSupporter:createDynList
1.5.5.1.1.9.1	The DMS's port managers within the list shall be hidden by default.	DMS	View DMS List	DMSListSupporter:createDynList

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.5.1.1.10	The detailed data displayed for a DMS shall include the connection site name (i.e., the name of the server from which the object is served).	DMS	View DMS List	chartlite.servlet.dms_dynlist_classes, DMSListSupporter:createDynList
1.5.5.1.1.10.1	The DMS connection site name within the list shall be hidden by default.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.1.11	The detailed data displayed for a DMS shall include an indicator of whether or not a DMS's travel time schedule is overridden.	DMS	View DMS List	chartlite.servlet.dms_dynlist_classes, DMSListSupporter:createDynList
1.5.5.1.1.11.1	The DMS travel time schedule overridden indicator within the list shall be hidden by default.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.1.12	The detailed data displayed for a DMS shall include the State Milepost where the DMS is located (if specified).	DMS	View DMS List	chartlite.servlet.dms_dynlist_classes, DMSListSupporter:createDynList
1.5.5.1.1.12.1	The State Milepost for a DMS within the list shall be hidden by default.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.1.13	The detailed data displayed for a DMS shall include the Owning Organization of the device.	DMS	View DMS List	chartlite.servlet.dms_dynlist_classes, DMSListSupporter:createDynList
1.5.5.1.1.13.1	The Owning Organization for a DMS within the list shall be hidden by default.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.2	The system shall allow the user to sort the list of DMSs that exist in the system.	EXISTING	View DMS List	DMSListSupporter:createDynList
1.5.5.1.2.6	The system shall allow the user to sort the list of DMSs by County.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.2.7	The system shall allow the user to sort the list of DMSs by Route.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.2.8	The system shall allow the user to sort the list of DMSs by Direction.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.2.9	The system shall allow the user to sort the list of DMSs by Port Manager name.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.2.10	The system shall allow the user to sort the list of DMSs by Connection Site name.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.2.11	The system shall allow the user to sort the list of DMSs by Travel Time schedule overridden indicator.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.2.12	The system shall allow the user to sort the list of DMSs by State Milepost.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.2.13	The system shall allow the user to sort the list of DMSs by Owning Organization.	DMS	View DMS List	DMSListSupporter:createDynList

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.5.1.3	The system shall allow the user to filter the list of DMSs that exist in the system.	EXISTING	View DMS List	DMSListSupporter:createDynList
1.5.5.1.3.4	The system shall allow the user to filter the list of DMSs by County.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.3.5	The system shall allow the user to filter the list of DMSs by Route.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.3.6	The system shall allow the user to filter the list of DMSs by Direction.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.3.7	The system shall allow the user to filter the list of DMSs by Port Manager name.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.3.8	The system shall allow the user to filter the list of DMSs by Connection Site name.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.3.9	The system shall allow the user to filter the list of DMSs by Owning Organization.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.3.10	The system shall allow the user to filter the list of DMSs by the Travel Time Schedule Overridden indicator (overridden, not overridden, or not applicable)	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.3.11	The system shall allow the user to filter the list of DMSs to include/exclude external DMSs, if the user has the right to view external DMSs.	DMS	View DMS List	chartlite.servlet.dms.dynlist_classes, DMSListSupporter:createDynList
1.5.5.1.3.11.1	The system shall automatically filter out external DMSs if the user does not have the right to view external DMSs.	DMS	View DMS List	DMSListSupporter:createDynList, DMSControlClassDiagram-ExternalDMS CD
1.5.5.1.3.12	The system shall allow the user to filter the list of DMSs to include/exclude internal (CHART) DMSs, if the user has the right to view external DMSs.	DMS	View DMS List	DMSListSupporter:createDynList
1.5.5.1.3.13	When the DMS list is filtered such that external DMSs are included, the system shall allow the user to filter the list to include/exclude external devices on an agency by agency basis. (For example, view VDOT DMSs, but not DDOT DMSs).	DMS	View DMS List	DMSListSupporter:createDynList, DMSControlClassDiagram-ExternalDMS CD, ExternalDMS CD
1.5.5.2	The system shall allow the user to view the list of HARs that exist in the system.	EXISTING	View HAR List	HARListSupporter:createDynList
1.5.5.2.1	The system shall allow the user to view detailed data for each HAR in the list	EXISTING	N/A	HARListSupporter:createDynList
1.5.5.2.1.7	The detailed data displayed for a HAR shall include the County where the HAR is located (if specified).	HAR	View HAR List	HARListSupporter:createDynList

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.5.2.1.8	The detailed data displayed for a HAR shall include the Route where the HAR is located (if specified).	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.1.9	The detailed data displayed for a HAR shall include the Direction of the route where the HAR is located (if specified).	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.1.10	The detailed data displayed for a HAR shall include the Port Managers assigned to a HAR.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.1.10.1	The HAR's port managers within the list shall be hidden by default.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.1.11	The detailed data displayed for a HAR shall include the connection site name (i.e., the name of the server from which the object is served).	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.1.11.1	The HAR connection site name within the list shall be hidden by default.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.1.12	The detailed data displayed for a HAR shall include the State Milepost where the HAR is located (if specified).	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.1.12.1	The State Milepost for a HAR within the list shall be hidden by default.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.1.13	The detailed data displayed for a HAR shall include the Owning Organization of the device.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.1.13.1	The Owning Organization for a HAR within the list shall be hidden by default.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.2	The system shall allow the user to sort the list of HARs that exist in the system.	EXISTING	View HAR List	HARListSupporter:createDynList
1.5.5.2.2.7	The system shall allow the user to sort the list of HARs by County.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.2.8	The system shall allow the user to sort the list of HARs by Route.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.2.9	The system shall allow the user to sort the list of HARs by Direction.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.2.10	The system shall allow the user to sort the list of HARs by Port Manager name.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.2.11	The system shall allow the user to sort the list of HARs by Connection Site name.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.2.12	The system shall allow the user to sort the list of HARs by State Milepost.	HAR	View HAR List	HARListSupporter:createDynList

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.5.2.2.13	The system shall allow the user to sort the list of HARs by Owning Organization.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.3	The system shall allow the user to filter the list of HARs that exist in the system.	EXISTING	View HAR List	HARListSupporter:createDynList
1.5.5.2.3.5	The system shall allow the user to filter the list of HARs by County.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.3.6	The system shall allow the user to filter the list of HARs by Route.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.3.7	The system shall allow the user to filter the list of HARs by Direction.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.3.8	The system shall allow the user to filter the list of HARs by Port Manager name.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.3.9	The system shall allow the user to filter the list of HARs by Connection Site name.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.2.3.10	The system shall allow the user to filter the list of HARs by Owning Organization.	HAR	View HAR List	HARListSupporter:createDynList
1.5.5.3	The system shall allow the user to view the list of SHAZAMs that exist in the system.	EXISTING	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.1	The system shall allow the user to view detailed data for each SHAZAM in the list	EXISTING	N/A	SHAZAMListSupporter:createDynList
1.5.5.3.1.7	The detailed data displayed for a SHAZAM shall include the County where the SHAZAM is located (if specified).	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.1.8	The detailed data displayed for a SHAZAM shall include the Route where the SHAZAM is located (if specified).	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.1.9	The detailed data displayed for a SHAZAM shall include the Direction of the route where the SHAZAM is located (if specified).	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.1.10	The detailed data displayed for a SHAZAM shall include the Port Managers assigned to a SHAZAM.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.1.10.1	The SHAZAM's port managers within the list shall be hidden by default.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.1.11	The detailed data displayed for a SHAZAM shall include the connection site name (i.e., the name of the server from which the object is served).	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.1.11.1	The SHAZAM connection site name within the list shall be hidden by default.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.5.3.1.12	The detailed data displayed for a SHAZAM shall include the State Milepost where the SHAZAM is located (if specified).	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.1.12.1	The State Milepost for a SHAZAM within the list shall be hidden by default.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.1.13	The detailed data displayed for a SHAZAM shall include the Owning Organization of the device.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.1.13.1	The Owning Organization for a SHAZAM within the list shall be hidden by default.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.2	The system shall allow the user to sort the list of SHAZAMs that exist in the system.	EXISTING	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.2.7	The system shall allow the user to sort the list of SHAZAMs by County.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.2.8	The system shall allow the user to sort the list of SHAZAMs by Route.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.2.9	The system shall allow the user to sort the list of SHAZAMs by Direction.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.2.10	The system shall allow the user to sort the list of SHAZAMs by Port Manager name.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.2.11	The system shall allow the user to sort the list of SHAZAMs by Connection Site name.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.2.12	The system shall allow the user to sort the list of SHAZAMs by State Milepost.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.2.13	The system shall allow the user to sort the list of SHAZAMs by Owning Organization.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.3	The system shall allow the user to filter the list of SHAZAMs that exist in the system.	EXISTING	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.3.3	The system shall allow the user to filter the list of SHAZAMs by County.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.3.4	The system shall allow the user to filter the list of SHAZAMs by Route.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.3.5	The system shall allow the user to filter the list of SHAZAMs by Direction.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.3.6	The system shall allow the user to filter the list of SHAZAMs by Port Manager name.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.3.3.7	The system shall allow the user to filter the list of SHAZAMs by Connection Site name.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.5.3.3.8	The system shall allow the user to filter the list of SHAZAMs by Owning Organization.	SHAZAM	View SHAZAM list	SHAZAMListSupporter:createDynList
1.5.5.4	The system shall allow the user to view the list of Detectors that exist in the system.	EXISTING	View Detector List	TSSListSupporter:createDynList
1.5.5.4.1	The system shall allow the user to view detailed data for each Detector in the list	EXISTING	N/A	TSSListSupporter:createDynList
1.5.5.4.1.6	The detailed data displayed for a Detector shall include the County where the Detector is located (if specified).	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.1.7	The detailed data displayed for a Detector shall include the Route where the Detector is located (if specified).	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.1.8	The detailed data displayed for a Detector shall include the Direction of the route where the Detector is located (if specified).	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.1.9	The detailed data displayed for a Detector shall include the Port Managers assigned to a Detector.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.1.9.1	The Detector's port managers within the list shall be hidden by default.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.1.10	The detailed data displayed for a Detector shall include the connection site name (i.e., the name of the server from which the object is served).	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.1.10.1	The Detector connection site name within the list shall be hidden by default.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.1.11	The detailed data displayed for a Detector shall include the State Milepost where the Detector is located (if specified).	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.1.11.1	The State Milepost for a Detector within the list shall be hidden by default.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.1.12	The detailed data displayed for a Detector shall include the Owning Organization of the detector.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.1.12.1	The Owning Organization for a Detector within the list shall be hidden by default.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.1.13	The system shall allow the user to view additional details for each detector shown in the Detector list. [NEW BUT ALREADY IMPLEMENTED]	TSS	N/A	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.5.4.1.13.1	The system shall allow a suitably privileged user to view the volume, speed, and occupancy data for the zone groups configured for a detector. [NEW BUT ALREADY IMPLEMENTED]	TSS	N/A	
1.5.5.4.1.13.1.1	The speed shown shall be an exact speed or a speed range depending on the user's rights.	TSS	View Detector Details, Configure TSS Speed Summary Ranges	GUITSSDataClasses
1.5.5.4.1.13.1.2	The system shall allow a suitably privileged user to view the volume, speed, and occupancy data for each zone included in a zone group.	TSS	View Lane Specific Data	GUITSSDataClasses, TSSManagementModulePkg CD
1.5.5.4.2	The system shall allow the user to sort the list of Detectors that exist in the system.	EXISTING	View Detector List	TSSListSupporter:createDynList
1.5.5.4.2.6	The system shall allow the user to sort the list of Detectors by County.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.2.7	The system shall allow the user to sort the list of Detectors by Route.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.2.8	The system shall allow the user to sort the list of Detectors by Direction.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.2.9	The system shall allow the user to sort the list of Detectors by Port Manager name.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.2.10	The system shall allow the user to sort the list of Detectors by Connection Site name.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.2.11	The system shall allow the user to sort the list of Detectors by State Milepost.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.2.12	The system shall allow the user to sort the list of Detectors by Owning Organization.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.3	The system shall allow the user to filter the list of Detectors that exist in the system.	EXISTING	View Detector List	TSSListSupporter:createDynList
1.5.5.4.3.3	The system shall allow the user to filter the list of Detectors by County.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.3.4	The system shall allow the user to filter the list of Detectors by Route.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.3.5	The system shall allow the user to filter the list of Detectors by Direction.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.3.6	The system shall allow the user to filter the list of Detectors by Port Manager name.	TSS	View Detector List	TSSListSupporter:createDynList

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.5.4.3.7	The system shall allow the user to filter the list of Detectors by Connection Site name.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.3.8	The system shall allow the user to filter the list of Detectors by Owning Organization.	TSS	View Detector List	TSSListSupporter:createDynList
1.5.5.4.3.9	The system shall allow the user to filter the list of Detectors to include/exclude external Detectors, if the user has the right to view external Detectors.	TSS	View Detector List	TSSListSupporter:createDynList, TSSManagement CD
1.5.5.4.3.9.1	The system shall automatically filter out external Detectors if the user does not have the right to view external Detectors.	TSS	View Detector List	TSSListSupporter:createDynList, TSSManagement CD
1.5.5.4.3.10	The system shall allow the user to filter the list of Detectors to include/exclude internal (CHART) Detectors, if the user has the right to view external Detectors.	TSS	View Detector List	TSSListSupporter:createDynList, TSSManagement CD
1.5.5.4.3.11	When the Detector list is filtered such that external Detectors are included, the system shall allow the user to filter the list to include/exclude external devices on an agency by agency basis. (For example, view VDOT Detectors, but not DDOT Detectors).	TSS	View Detector List	TSSListSupporter:createDynList, TSSManagement CD
1.5.5.5	The system shall allow the user to view the list of Cameras that exist in the system.	EXISTING	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.1	The system shall allow the user to view detailed data for each Camera in the list	EXISTING	N/A	VideoSourceListSupporter:createDynList
1.5.5.5.1.6	The detailed data displayed for a Camera shall include the County where the Camera is located (if specified).	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.1.7	The detailed data displayed for a Camera shall include the Route where the Camera is located (if specified).	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.1.8	The detailed data displayed for a Camera shall include the Direction of the route where the Camera is located (if specified).	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.1.9	The detailed data displayed for a Camera shall include the State Milepost where the Camera is located (if specified).	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.1.9.1	The State Milepost for a Camera within the list shall be hidden by default.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.1.10	The detailed data displayed for a Camera shall include the Owning Organization of the camera.	Video	View Camera List	VideoSourceListSupporter:createDynList

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.5.5.1.10.1	The Owning Organization for a Camera within the list shall be hidden by default.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.1.11	The detailed data displayed for a Camera shall include the Connection Site name.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.1.11.1	The Connection Site name for a Camera within the list shall be hidden by default.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.2	The system shall allow the user to sort the list of Cameras that exist in the system.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.2.6	The system shall allow the user to sort the list of Cameras by County.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.2.7	The system shall allow the user to sort the list of Cameras by Route.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.2.8	The system shall allow the user to sort the list of Cameras by Direction.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.2.9	The system shall allow the user to sort the list of Cameras by State Milepost.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.2.10	The system shall allow the user to sort the list of Cameras by Owning Organization.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.2.11	The system shall allow the user to sort the list of Cameras by Connection Site name.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.3	The system shall allow the user to filter the list of Cameras that exist in the system.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.3.4	The system shall allow the user to filter the list of Cameras by County.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.3.5	The system shall allow the user to filter the list of Cameras by Route.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.3.6	The system shall allow the user to filter the list of Cameras by Direction.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.3.7	The system shall allow the user to filter the list of Cameras by Owning Organization.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.5.3.8	The system shall allow the user to filter the list of Cameras by Connection Site name.	Video	View Camera List	VideoSourceListSupporter:createDynList
1.5.5.6	The system shall allow the user to view the list of Monitors that exist in the system.	Video	View Monitor List	MonitorListSupporter:createDynList
1.5.5.6.1	The system shall allow the user to view detailed data for each Monitor in the list	Video	N/A	MonitorListSupporter:createDynList
1.5.5.5.1.6	The detailed data displayed for a Monitor shall include the Connection Site name.	Video	View Monitor List	MonitorListSupporter:createDynList

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
1.5.5.5.1.6.1	The Connection Site name for a Monitor within the list shall be hidden by default.	Video	View Monitor List	MonitorListSupporter:createDynList
1.5.5.6.2	The system shall allow the user to sort the list of Monitors that exist in the system.	Video	View Monitor List	MonitorListSupporter:createDynList
1.5.5.6.2.6	The system shall allow the user to sort the Monitor list by Connection Site name.	Video	View Monitor List	MonitorListSupporter:createDynList
1.5.5.6.3	The system shall allow the user to filter the list of Monitors that exist in the system.	Video	View Monitor List	MonitorListSupporter:createDynList
1.5.5.6.3.5	The system shall allow the user to filter the Monitor list by Connection Site name.	Video	View Monitor List	MonitorListSupporter:createDynList
1.5.5.7	The system shall allow the user to choose the columns to display in a device list.	Device	View DMS List, View Monitor List, View HAR List, View SHAZAM List, View Camera List, View Monitor List, Set Column Visibility	DynListReqHdlrDelegate.createDynList, setColumnVisibility, DynamicListClasses
1.5.5.7.1	The column containing the device name shall always be displayed.	Device	Set Column Visibility	DMSListSupporter.createDynList HARListSupporter.createDynList SHAZAMListSupporter.createDynList TSSListSupporter.createDynList VideoSourceListSupporter.createDynList, MonitorListSupporter:createDynList,
1.5.5.7.2	The system shall store the user's selection for columns to be displayed and initially show only the selected columns the next time the list is shown.	Device	Set Column Visibility	DynListReqHdlrDelegate.createDynList, setColumnVisibility, DynamicListClasses
1.5.5.7.2.1	The user's column display selection shall be stored on a per-user basis on the computer currently in use by the user. (A browser cookie)	Device	Set Column Visibility	DynListReqHdlrDelegate.createDynList, setColumnVisibility, DynamicListClasses
1.5.5.7.2.2	The user's column display selection shall be per device list. (The settings for the DMS list can be different than those for the HAR list, etc.)	Device	Set Column Visibility	DynListReqHdlrDelegate.createDynList, setColumnVisibility, DynamicListClasses
3	MONITOR TRAFFIC AND ROADWAYS	HEADER	N/A	
3.2	RECORD CONDITIONS	HEADER	N/A	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
3.2.1	The system shall capture data from internal CHART detectors, sensors, and probes that automatically send their data to the CHART database when they are polled or send their signal. *	EXISTING	N/A	
3.2.2	The system shall automatically send the recorded data from internal CHART detectors and sensors to the archive after the pre-determined time. *	EXISTING	N/A	
3.2.2.1	The system shall archive all detector traffic monitoring data per detection zone. *	EXISTING	N/A	
3.3	ISSUE ALERT OR POST INFORMATION	HEADER	N/A	
3.3.10	The system shall prevent duplicate, non-closed alerts from being displayed to users.	EXISTING	N/A	
3.3.10.7	Two External Connection Alerts shall be considered duplicates when the external connection they reference are the same.	External Interface	Confirm Unique Alert	compare() in ExternalConnectionAlertImpl in AlertModule CD
3.3.10.8	Two External Event Alerts shall be considered duplicates when the external event they reference are the same.	External Interface	Confirm Unique Alert	compare() in ExternalEventAlertImpl in AlertModule CD
3.3.10.9	Two Travel Time Alerts shall be considered duplicates when the travel route they reference are the same.	Travel Times	Confirm Unique Alert	compare() in TravelTimeAlertImpl in AlertModule CD
3.3.10.10	Two Toll Rate Alerts shall be considered duplicates when the travel route they reference are the same and the alert description text is identical.	Toll Rates	Confirm Unique Alert	compare() in TollRateAlertImpl in AlertModule CD
3.4	RECEIVE AND RESPOND TO ALERT	HEADER	N/A	
3.4.1.	A suitably privileged operator shall be able to view alerts.*	EXISTING	View Alerts	viewAlertsInitialView, viewAlertsPeriodicUpdate
3.4.1.1.	The system shall display alerts at all times on the CHART home page.	EXISTING	View Alerts	viewAlertsInitialView, viewAlertsPeriodicUpdate
3.4.1.1.3	The system shall provide a visual cue to the user when there are alerts in the “New” state.	EXISTING	View Alerts	viewNewAlerts
3.4.1.1.4	The system shall provide an audio cue to the user when there are alerts in the “New” state.	EXISTING	View Alerts	getAlertSound
3.4.1.1.4.1	The system shall allow an administrator to configure the sound that is played as the audio cue, per alert type.	EXISTING	Configure R3B3 Alert Types	ConfigureAlertAudioCue

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
3.4.1.1.4.3	The systems shall play the corresponding audio cue (by alert type) to the user upon receipt by the user session of the new alert.	EXISTING	View Alerts	viewNewAlerts
3.4.2.	A suitably privileged user shall be able to manage alerts through the following states: New, Accepted, Delayed, and Closed.	EXISTING	R3B3ManageAlertsAndNotifications diagram	performAlertAction
3.4.2.6	A suitably privileged user shall be able to Resolve alerts in the New, Accepted, and Delayed states	EXISTING	Resolve Alert	resolveAlert
3.4.2.6.9	Clicking the Resolve link of an External Connection Alert shall cause the external connection status list to be displayed.	External Interface	Resolve Alert	resolveAlert
3.4.2.6.10	Clicking the Resolve link of an External Event Alert shall cause the details page for the external event to be displayed.	External Interface	Resolve Alert	resolveAlert
3.4.2.6.11	Clicking the Resolve link of a Travel Time Alert shall cause the details page for the associated Travel Route to be displayed.	Travel Times	Resolve Alert	resolveAlert
3.4.2.6.12	Clicking the Resolve link of a Toll Rate Alert shall cause the details page for the associated Travel Route to be displayed.	Toll Rates	Resolve Alert	resolveAlert
4	MANAGE EVENTS	HEADER	N/A	
4.2	OPEN EVENT	HEADER	N/A	
4.2.1	The system shall allow a suitably privileged user to create a new event.	EXISTING	N/A	
4.2.1.3	The system shall store the associated Center with each event entry.	EXISTING	N/A	
4.2.1.3.2	The system shall support the transfer of responsibility for an open event from one Center to another.	EXISTING	Transfer Traffic Event	
4.2.1.3.2.1	When a traffic event is transferred to a new operations center, the system shall reassign the owning organization to the owning organization of the operations center to which the traffic event has been transferred.	Public / Private Data Sharing	Transfer Traffic Event	
4.2.1.8	The system shall assign a Traffic Event an owning organization based on the owning organization of the operations center the creating user has logged into.	Public Private Data Sharing	Create Traffic Event 1	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
4.2.3	DEPLOY RESOURCES The system shall allow the user to view the pre-defined decision support plans to suggest the course of action and notifications, and execute the selected (or modified) course of action. The ability to record the deploying of resources only applies to user generated events – not External Events.*	FUTURE	N/A	
4.2.3.2	EVALUATE EVENT RESPONSE RECOMMENDATIONS. The system shall display the most appropriate corresponding recommended response plan from the pre-defined decision support plans, based on the event type, conditions, day of week and time of day (e.g., to determine closest open maintenance shop), location, and area of responsibility.	FUTURE	N/A	
4.2.3.2.1	The system shall display the recommended DMS, HAR, Detector, Detectpr. SHAZAM, CCTV camera and monitor usage and the corresponding message/control, based on the event location *	Close Devices	View Devices Near Event	TrafficEventReqHdlr: getNearbyDevicesJSON, getNearbyCameraJSONArray, getNearbyDMSJSONArray, getNearbyHARJSONArray, getNearbyTSSJSONArray, ServletUtil:getNearbyObjects
4.2.3.2.1.1	The recommended device usage will be available only if the traffic event has a latitude and longitude specified.	Close Devices	View Devices Near Event	getNearbyDevicesJSON
4.2.3.2.1.2	The system shall display all applicable devices located within a specified radius of the traffic event.	Close Devices	View Devices Near Event, Set Close Device Radius	getNearbyDevicesJSON
4.2.3.2.1.2.1	Devices that do not have a lat/long specified will not be displayed.	Close Devices	View Devices Near Event	getNearbyDevicesJSON
4.2.3.2.1.2.2	The user shall be permitted to change the radius used to determine if devices should be recommended based on their location.	Close Devices	Set Close Device Radius	getNearbyDevicesJSON
4.2.3.2.1.2.2.1	The user's current radius setting for each event shall be retained until the user logs out.	Close Devices	Set Close Device Radius	getNearbyDevicesJSON
4.2.3.2.1.2.3	The system shall include both CHART and external devices as applicable that are located within the specified radius of the traffic event.	Close Devices	View Devices Near Event	getNearbyDevicesJSON

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
4.2.3.2.1.2.4	The system shall display details for each recommended DMS as available, including Name/Location Description, Distance from the Event, Route, Direction, Intersecting Feature, Current Message, Current Beacon State, Current Mode (online/offline/maintenance), and current status.	Close Devices	View Devices Near Event	getNearbyDMSJSONArray
4.2.3.2.1.2.5	The system shall display the details for each recommended HAR as available, including Name/Location Description, Distance from the Event, Route, Direction, Intersecting Feature, Current Mode, Current Status, and Current Transmitter Status.	Close Devices	View Devices Near Event	getNearbyHARJSONArray
4.2.3.2.1.2.6	The system shall display the details for each recommended Detector as available, including Name/Location Description, Distance from the Event, Route, Direction, Intersecting Feature, Current Mode, Current Status, and Average Speed.	Close Devices	View Devices Near Event	getNearbyTSSJSONArray
4.2.3.2.1.2.6.1	The display of Average Speed for a detector shall be subject to the user's rights to view detailed detector VSO data OR view summary detector VSO data for the organization which owns the device. (User sees actual Average Speed, Average Speed Range, or No Average Speed depending on their user rights).	Close Devices	View Devices Near Event	getNearbyTSSJSONArray
4.2.3.2.1.2.7	The system shall display the details for each recommended Camera as available, including Name/Location Description, Distance from Event, Route, Direction, Intersecting Feature, Current Status, Local Monitors where it is currently displayed, and the Current Controlling Op Center.	Close Devices	View Devices Near Event	getNearbyCameraJSONArray
4.2.3.2.1.3	The system shall allow a user with proper rights to choose to add a recommended DMS to the response plan of the event.	Close Devices	Add Close DMS to Response Plan	(simple variant of existing code, no design req'd)
4.2.3.2.1.3.1	The system shall add the DMS to the response plan with an empty message specified.	Close Devices	Add Close DMS to Response Plan	(simple variant of existing code, no design req'd)
4.2.3.2.1.3.2	The system shall not permit an external DMS to be added to the response plan of the event.	Close Devices	Add Close DMS to Response Plan	(simple variant of existing code, no design req'd)
4.2.3.2.1.4	The system shall allow a user with proper rights to choose to add a recommended HAR to the response plan of the event.	Close Devices	Add Close HAR to Response Plan	(simple variant of existing code, no design req'd)

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
4.2.3.2.1.4.1	The system shall add the HAR to the response plan with an empty message specified. (default header, empty body, default trailer)	Close Devices	Add Close HAR to Response Plan	(simple variant of existing code, no design req'd)
4.2.3.3	SELECT/ MODIFY COURSE OF ACTION The system shall allow the user to accept, modify, or bypass the decision support recommendations for device usage, message, and control; for resource requests and notifications; for equipment type; and for equipment location.	FUTURE	N/A	
4.2.3.3.7	SELECT DEVICE PLAN OR PLAN ITEMS	HEADER	N/A	
4.2.3.4	EXECUTE COURSE OF ACTION	HEADER	N/A	
4.2.3.4.7	The system shall automatically send out the selected messages or notifications to the specified resources in accordance with the selected course of action.	FUTURE	N/A	
4.2.3.4.7.1	The system shall provide the capability to automatically notify individuals of specific system events.	FUTURE	N/A	
4.2.4	The system shall allow a suitably privileged user to create a CHART Event from an external event.	External Interface	Create CHART Event from External Event	copyExternalEventAsCHARTEventWithoutForm
4.2.4.1	When a CHART Event is created from an external event the system shall automatically associate the CHART Event with the external event.	External Interface	Create CHART Event from External Event, Associate CHART Event with External Event	copyExternalEventAsCHARTEventWithoutForm
4.3	RESPOND TO AND MONITOR EVENT	HEADER	N/A	
4.3.1	MONITOR EVENT. The system shall allow the user to view and update the status of devices, resources responding to an event, and the event response activities.	EXISTING	N/A	
4.3.1.3	MONITOR DEVICE STATUS	HEADER	N/A	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
4.3.1.3.6	For devices that were already displaying a message before an event was initiated, the system shall automatically arbitrate the message queue in accordance with the previously defined business rules. (Suggestion for validation: there may need to be some adjustments to the message protocols and arbitration business rules to address prioritizing messages based on geography and severity in addition to just event type (which is currently the only arbitration queue factor). Additional suggestion: Ensure that travel time messages do not interfere with incident-related messages.) *	EXISTING	Add Message To Device Queue	
4.3.1.3.6.1	Messages shall be ordered in a queue based on their priority.	EXISTING	Add Message To Device Queue	
4.3.1.3.6.1.1	A message entering a device queue with a higher priority than the current message on the device shall preempt the current message (except where combination rules apply).	EXISTING	Evaluate DMS Device Queue Entries	
4.3.1.3.6.1.3	A preempted message shall be returned to the device queue.	EXISTING	Evaluate DMS Device Queue Entries	
4.3.1.3.6.1.3.1	The system shall log a message to the operations log when a message is preempted.	EXISTING	Evaluate DMS Device Queue Entries	
4.3.1.3.6.2	The system shall support a method of specifying which message types may be combined in a DMS queue.	EXISTING	Evaluate DMS Device Queue Entries, Configure DMS Message Combination Rules	
4.3.1.3.6.3	The system shall allow a suitably privileged user to select combinations of different message types in a DMS Queue.	EXISTING	Prioritize Device Queue	
4.3.1.3.6.5	The system shall maintain the identity of each unique message in a combined message such that the process of creating the combined message can be reversed.	EXISTING	Evaluate DMS Device Queue Entries	
4.3.1.3.6.6	The system shall support the concatenation of messages in a queue on a single device for simultaneous display/broadcast.	EXISTING	Evaluate DMS Device Queue Entries	
4.3.1.3.6.7	The system shall maintain attributes for messages in a device queue.	EXISTING	Add Message To Device Queue	
4.3.1.3.6.7.1	Device queue message attributes shall include the event the message is associated with.	EXISTING	Add Message To Device Queue	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
4.3.1.3.6.7.2	Device queue message attributes shall include the Center responsible for the message.	EXISTING	Add Message To Device Queue	
4.3.1.3.6.8	The system shall support the assignment of priorities to queued messages based on the message type/source.	EXISTING	Add Message To Device Queue	
4.3.1.3.6.8.4	The system shall support an arbitration queue priority level of Toll Rate used for messages informing drivers of the toll rate to a given destination.	Toll Rates	Add Message To Device Queue	ArbQueuePriorityLevel in SystemInterfaces/DeviceManagement
4.3.1.3.6.8.4.1	The Toll Rate level shall have a relative priority level of 7. (Note – this number is only used to describe its relationship to other levels in the requirements but does not necessarily imply a numeric implementation).	Toll Rates	Add Message To Device Queue	ArbQueuePriorityLevel in SystemInterfaces/DeviceManagement
4.3.1.3.6.8.5	The system shall support an arbitration queue priority level of Travel Time used for messages informing drivers of the travel time to a given destination.	Travel Times	Add Message To Device Queue	ArbQueuePriorityLevel in SystemInterfaces/DeviceManagement
4.3.1.3.6.8.5.1	The Travel Time level shall have a relative priority level of 6. (Note – this number is only used to describe its relationship to other levels in the requirements but does not necessarily imply a numeric implementation).	Travel Times	Add Message To Device Queue	ArbQueuePriorityLevel in SystemInterfaces/DeviceManagement
4.3.5	VIEW EVENT LIST	HEADER	N/A	
4.3.5.1	The system shall allow the user to view a list of Open events in the system. [NEW BUT ALREADY IMPLEMENTED]	EXISTING	N/A	
4.3.5.1.1	The system shall show the Description of the event [NEW BUT ALREADY IMPLEMENTED]	EXISTING	N/A	
4.3.5.1.1.1	The system shall allow the user to sort the list by Description. [NEW BUT ALREADY IMPLEMENTED]	EXISTING	N/A	
4.3.5.1.2	The system shall show the Location Description of the event. [NEW BUT ALREADY IMPLEMENTED]	EXISTING	N/A	
4.3.5.1.2.1	The system shall allow the user to sort the list by Location Description. [NEW BUT ALREADY IMPLEMENTED]	EXISTING	N/A	
4.3.5.1.3	The system shall show the Time Opened for an event.	Misc	View Time Opened	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
4.3.5.1.3.1	The Time Opened shall be hidden in the list by default.	Misc	View Time Opened	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol
4.3.5.1.3.2	The system shall allow the user to sort the list by Time Opened.	Misc	View Time Opened	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol
4.3.5.1.3.3	The system shall allow the user to filter the list by Time Opened.	Misc	View Time Opened	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol
4.3.5.1.4	The system shall show the Time Last Modified for an event.	Misc	View Time Last Modified	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol
4.3.5.1.4.1	The Time Last Modified shall be hidden in the list by default.	Misc	View Time Last Modified	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol
4.3.5.1.4.2	The system shall allow the user to sort the list by Time Last Modified.	Misc	View Time Last Modified	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol
4.3.5.1.4.3	The system shall allow the user to filter the list by Time Last Modified.	Misc	View Time Last Modified	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol
4.3.5.1.5	The system shall show the Percent Of Lanes Closed for an event.	Misc	View Lane Closed Percent	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol
4.3.5.1.5.1	The Percent Of Lanes Closed shall be hidden in the list by default.	Misc	View Lane Closed Percent	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol
4.3.5.1.5.2	The system shall allow the user to sort the list by Percent Of Lanes Closed.	Misc	View Lane Closed Percent	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol
4.3.5.1.5.3	The system shall allow the user to filter the list by Percent Of Lanes Closed.	Misc	View Lane Closed Percent	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol
4.3.5.1.6	The system shall show the Network Connection Site for an event.	Misc	View Network Connection Site	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
4.3.5.1.6.1	The Network Connection Site shall be hidden in the list by default.	Misc	View Network Connection Site	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol
4.3.5.1.6.2	The system shall allow the user to sort the list by Network Connection Site.	Misc	View Network Connection Site	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol
4.3.5.1.6.3	The system shall allow the user to filter the list by Network Connection Site.	Misc	View Network Connection Site	TrafficEventDynListSupporter:createDynList, createDynListCols, addCol
4.3.5.4	The system shall allow the user to choose the columns to display in a traffic event list.	Misc	View Traffic Event List, Set Column Visibility	TrafficEventDynListSupporter.createDynListCols
4.3.5.4.1	The column containing the event name shall always be displayed.	Misc	Set Column Visibility	TrafficEventDynListSupporter.createDynListCols
4.3.5.4.2	The system shall store the user's selection for columns to be displayed and initially show only the selected columns the next time the list is shown.	Misc	Set Column Visibility	TrafficEventDynListSupporter.createDynListCols
4.3.5.4.2.1	The user's column display selection shall be stored on a per-user basis on the computer currently in use by the user. (A browser cookie)	Misc	Set Column Visibility	TrafficEventDynListSupporter.createDynListCols
4.3.5.4.2.2	The user's column display selection shall be per list type. (The settings for the Open Events list can be different than those for the Pending Events list, etc.)	Misc	Set Column Visibility	TrafficEventDynListSupporter.createDynListCols
5	MANAGE TRAFFIC	HEADER	N/A	N/A
5.3	CALCULATE TRAVEL TIMES (see also 2.3.2)	HEADER	N/A	N/A
5.3.6	The system shall calculate the travel time for a travel route that contains one or more roadway links.	Travel Times / Toll Rates	Compute Route Travel Time	TravelRouteFactoryImpl:updateLinkData, TravelRouteFactoryImpl:computeTravelTime
5.3.6.1	The system shall compute the travel time for a travel route by summing the travel time calculations for each of the travel route's roadway links.	Travel Times / Toll Rates	Compute Route Travel Time	TravelRouteFactoryImpl:computeTravelTime
5.3.6.1.1	The travel time for each roadway link shall be computed by multiplying the current travel time of the roadway link by a configured "percent included" setting for the roadway link. (This setting is specified in 5.4.1.5.4)	Travel Times / Toll Rates	Compute Route Travel Time	TravelRouteFactoryImpl:computeTravelTime

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.3.6.2	The system shall make the route travel time unavailable if the number of roadway links which do not meet its assigned quality level is greater than the number of links allowed to fall below their assigned quality level for the travel route. (These settings are discussed in 5.4.1.5.4.2 and 5.4.1.14)	Travel Times / Toll Rates	Compute Route Travel Time	TravelRouteFactoryImpl:computeTravelTime
5.3.6.3	The system shall issue Travel Time Alerts based on the computed travel time for a travel route using the settings as specified in 5.4.1.11.	Travel Times / Toll Rates	Compute Route Travel Time, Create Travel Time Alert	TravelRouteFactoryImpl:computeTravelTime
5.3.6.4	The system shall issue Travel Time Notifications based on the computed travel time for a travel route using the settings as specified in 5.4.1.11.	Travel Times / Toll Rates	Compute Route Travel Time, Send Travel Time Notification	TravelRouteFactoryImpl:computeTravelTime
5.4	MAINTAIN TRAVEL ROUTES. The system shall maintain “travel routes” for the purpose of managing and posting travel times and toll rates on those routes.	HEADER	N/A	TravelRouteModule
5.4.1	The system shall allow a suitably privileged user to add a travel route to the system.	Travel Times / Toll Rates	Add Travel Route	addEditTravelRouteForm, addEditTravelRoute, TravelRouteFactoryImpl:addRoute
5.4.1.1	The system shall require the user to specify a descriptive name for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, validateConfig in TravelRouteFactoryImpl:addRoute
5.4.1.2	The system shall require the user to specify a preferred destination name for the travel route for use in DMS messages.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, validateConfig in TravelRouteFactoryImpl:addRoute
5.4.1.3	The system shall allow the user to specify an alternate destination name for the travel route for use in DMS messages.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, TravelRouteFactoryImpl:addRoute
5.4.1.3.1	The system shall require that the alternate destination name (if entered) be shorter in length than the preferred destination name for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, validateConfig in TravelRouteFactoryImpl:addRoute
5.4.1.4	The system shall allow the user to specify a second alternate destination name for the travel route for use in DMS messages.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, validateConfig in TravelRouteFactoryImpl:addRoute
5.4.1.4.1	The system shall require that the second alternate destination name (if entered) be shorter in length than the first alternate destination name for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, validateConfig in TravelRouteFactoryImpl:addRoute

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.1.5	The system shall allow the user to specify the roadway links that comprise a travel route.	Travel Times / Toll Rates	Add Roadway Links To Travel Route, Edit Roadway Link Usage In Travel Route, Modify Roadway Link Order In Travel Route, Remove Roadway Link From Travel Route	addTravelRouteLinkForm, addTravelRouteLink, TravelRouteFactoryImpl:addRoute
5.4.1.5.1	The system shall allow the user to add a roadway link to the list of roadway links that comprise the travel route.	Travel Times / Toll Rates	Add Roadway Links To Travel Route	addTravelRouteLinkForm, addTravelRouteLink, TravelRouteFactoryImpl:addRoute
5.4.1.5.1.1	The system shall allow the user to select a roadway link from a list of all available roadway links.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, addOrUpdateLink, findTravelRouteLinksJSON
5.4.1.5.1.2	The system shall allow the user to filter the list of available roadway links.	Travel Times / Toll Rates	Specify Roadway Links To Add	findTravelRouteLinksJSON
5.4.1.5.1.2.1	County shall be an available roadway link filter criteria.	Travel Times / Toll Rates	Specify Roadway Links To Add	findTravelRouteLinksJSON
5.4.1.5.1.2.2	Route type shall be an available roadway link filter criteria.	Travel Times / Toll Rates	Specify Roadway Links To Add	findTravelRouteLinksJSON
5.4.1.5.1.2.3	Route name/number shall be an available roadway link filter criteria.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, addOrUpdateLink, findTravelRouteLinksJSON
5.4.1.5.1.2.4	Travel direction shall be an available roadway link filter criteria.	Travel Times / Toll Rates	Specify Roadway Links To Add	findTravelRouteLinksJSON
5.4.1.5.1.2.5	External ID shall be an available roadway link filter criteria.	Travel Times / Toll Rates	Specify Roadway Links To Add	findTravelRouteLinksJSON
5.4.1.5.1.3	The system shall allow the user to select one or more available roadway links to be added to the travel route.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, findTravelRouteLinksJSON
5.4.1.5.1.4	The system shall show details for each roadway link available for selection, to the extent that such data is provided and has been imported into the CHART system.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, findTravelRouteLinksJSON
5.4.1.5.1.4.1	The available roadway link details shall include the name of the external system that provides data for the roadway link.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, findTravelRouteLinksJSON, INRIXDefLinkImportProgramPkg:importINRIXLinks

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.1.5.1.4.2	The available roadway link details shall include the roadway link's external ID. (This is the ID by which the external system identifies the link.)	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, findTravelRouteLinksJSON, INRIXDefLinkImportProgramPkg:importINRIXLinks
5.4.1.5.1.4.3	The available roadway link details shall include the roadway link's name.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, findTravelRouteLinksJSON, INRIXDefLinkImportProgramPkg:importINRIXLinks
5.4.1.5.1.4.4	The available roadway link details shall include the name and/or number of the route where the roadway link is located.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, findTravelRouteLinksJSON, INRIXDefLinkImportProgramPkg:importINRIXLinks
5.4.1.5.1.4.5	The available roadway link details shall include the travel direction.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, findTravelRouteLinksJSON, INRIXDefLinkImportProgramPkg:importINRIXLinks
5.4.1.5.1.4.6	The available roadway link details shall include the length of the roadway link, in miles.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, findTravelRouteLinksJSON, INRIXDefLinkImportProgramPkg:importINRIXLinks
5.4.1.5.1.4.7	The available roadway link details shall include the county in which the roadway link is located.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, findTravelRouteLinksJSON, INRIXDefLinkImportProgramPkg:importINRIXLinks
5.4.1.5.1.4.8	The available roadway link details shall include the distance from the prior roadway link shown in the list, in miles.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, findTravelRouteLinksJSON
5.4.1.5.1.4.8.1	The distance from the prior roadway link shall be calculated as the distance from the end of the prior roadway link to the beginning of the roadway link.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, findTravelRouteLinksJSON, RoadwayLinkManager.suggestLinks
5.4.1.5.1.4.8.2	The distance from prior roadway link for the first available roadway link shall be the distance from the last roadway link currently included in the travel route if the travel route currently includes one or more roadway links.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, findTravelRouteLinksJSON
5.4.1.5.1.4.8.3	The distance from prior roadway link for the first available roadway link shall be N/A if there are no prior roadway links currently added to the travel route.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks, findTravelRouteLinksJSON
5.4.1.5.1.5	The system shall allow the user to view a list of suggested roadway links for selection if the travel route already contains one or more roadway links.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks
5.4.1.5.1.5.1	The list of suggested roadway links shall include the roadway link that has a beginning lat/long that is closest to the ending lat/long of the last roadway link currently included in the travel route.	Travel Times / Toll Rates	Specify Roadway Links To Add	suggestLinks

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.1.5.2	The system shall allow a user to remove a roadway link from the list of roadway links that comprise the travel route.	Travel Times / Toll Rates	Remove Roadway Link From Travel Route	removeTravelRouteLink
5.4.1.5.3	The system shall allow the user to specify the order in which the roadway links occur on the travel route.	Travel Times / Toll Rates	Modify Roadway Link Order In Travel Route	moveTravelRouteLink
5.4.1.5.4	The system shall allow the user to specify settings for the roadway links pertaining to their use within the travel route.	Travel Times / Toll Rates	Edit Roadway Link Usage In Travel Route, Specify Roadway Link Usage Settings	setTravelRouteLinkSettingsForm, setTravelRouteLinkSettings, TravelRouteFactoryImpl:addRoute
5.4.1.5.4.1	The system shall allow the user to set the percentage of the roadway link's travel time to be used when computing the travel route's travel time and distance.	Travel Times / Toll Rates	Specify Roadway Link Usage Settings	setTravelRouteLinkSettingsForm, setTravelRouteLinkSettings, TravelRouteFactoryImpl:addRoute
5.4.1.5.4.2	The system shall allow the user to set the minimum data quality allowed for the roadway link's data when using that data in the travel route.	Travel Times / Toll Rates	Specify Roadway Link Usage Settings	setTravelRouteLinkSettingsForm, setTravelRouteLinkSettings, TravelRouteFactoryImpl:addRoute
5.4.1.5.4.2.1	The system shall support a roadway link data quality level of Low.	Travel Times / Toll Rates	Specify Roadway Link Usage Settings	setTravelRouteLinkSettingsForm, setTravelRouteLinkSettings, TravelRouteFactoryImpl:addRoute
5.4.1.5.4.2.2	The system shall support a roadway link quality level of Medium.	Travel Times / Toll Rates	Specify Roadway Link Usage Settings	setTravelRouteLinkSettingsForm, setTravelRouteLinkSettings, TravelRouteFactoryImpl:addRoute
5.4.1.5.4.2.3	The system shall support a roadway link quality level of High.	Travel Times / Toll Rates	Specify Roadway Link Usage Settings	setTravelRouteLinkSettingsForm, setTravelRouteLinkSettings, TravelRouteFactoryImpl:addRoute
5.4.1.6	The system shall allow the user to view the roadway links that have been specified to comprise the travel route, including detailed data to the extent that such data is provided and has been imported into the CHART system.	Travel Times / Toll Rates	View Roadway Links Specified For Travel Route	addEditTravelRouteForm
5.4.1.6.1	The system shall display the system that supplies data for each roadway link.	Travel Times / Toll Rates	View Roadway Links Specified For Travel Route	addEditTravelRouteForm
5.4.1.6.2	The system shall display the ID of each roadway link as specified by the system that provides data for the roadway link.	Travel Times / Toll Rates	View Roadway Links Specified For Travel Route	addEditTravelRouteForm

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.1.6.3	The system shall display the name of each roadway link.	Travel Times / Toll Rates	View Roadway Links Specified For Travel Route	addEditTravelRouteForm
5.4.1.6.4	The system shall display the name and/or number of the route where each roadway link is located.	Travel Times / Toll Rates	View Roadway Links Specified For Travel Route	addEditTravelRouteForm
5.4.1.6.5	The system shall display the direction of traffic flow for each roadway link.	Travel Times / Toll Rates	View Roadway Links Specified For Travel Route	addEditTravelRouteForm
5.4.1.6.6	The system shall display the length of each roadway link.	Travel Times / Toll Rates	View Roadway Links Specified For Travel Route	addEditTravelRouteForm
5.4.1.6.7	The system shall display the county in which each roadway link is located.	Travel Times / Toll Rates	View Roadway Links Specified For Travel Route	addEditTravelRouteForm
5.4.1.6.8	The system shall display the distance from the beginning of each roadway link to the end of the prior link specified for the travel route.	Travel Times / Toll Rates	View Roadway Links Specified For Travel Route	addEditTravelRouteForm
5.4.1.6.8.1	The system shall not display a distance from the prior roadway link for the first roadway link in a travel route.	Travel Times / Toll Rates	View Roadway Links Specified For Travel Route	addEditTravelRouteForm
5.4.1.7	The system shall allow the user to specify a toll rate source for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	setTollRateSourceForm, setTollRateSource, TravelRouteFactoryImpl:addRoute
5.4.1.7.1	The system shall allow the user to select a toll rate source to provide toll rate data for the travel route.	Travel Times / Toll Rates	Select Toll Rate Source	setTollRateSourceForm, setTollRateSource
5.5.1.7.2	The system shall allow the user to remove the toll rate source for the travel route (effectively disabling toll rate data for the travel route)	Travel Times / Toll Rates	Remove Toll Rate Source	removeTollRateSource
5.4.1.8	The system shall require the user to specify the location data for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, TravelRouteFactoryImpl:addRoute
5.4.1.8.1	The system shall allow the user to choose to have the system derive the location data for the travel route from the roadway links that comprise the travel route, if the travel route contains 1 or more roadway links.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.8.2	The system shall allow the user to manually specify the location data for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.1.8.2.1	The system shall require the user to specify 1 or more counties in which the travel route is located.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, TravelRouteFactoryImpl:addRoute
5.4.1.8.2.2	The system shall require the user to specify 1 or more roads (routes) on which the travel route is located.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, TravelRouteFactoryImpl:addRoute
5.4.1.8.2.2.1	The system shall require the user to specify each route's route type.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, TravelRouteFactoryImpl:addRoute
5.4.1.8.2.2.2	The system shall require the user to specify each route's name or number.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, TravelRouteFactoryImpl:addRoute
5.4.1.8.2.2.3	The system shall require the user to specify each route's direction.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, TravelRouteFactoryImpl:addRoute
5.4.1.9	The system shall allow the user to enable travel times for the travel route if one or more roadway links are specified for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, TravelRouteFactoryImpl:addRoute
5.4.1.9.1	The system shall require the user to specify the maximum travel time for the travel route if travel times are enabled for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, TravelRouteFactoryImpl:addRoute
5.4.1.9.1.2	The system shall not display the route's travel time on any DMS if the route's travel time exceeds the specified maximum value.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TravelRouteImpl:computeTravelTime
5.4.1.9.2	The system shall require the user to specify the minimum travel time for the travel route if travel times are enabled for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.9.2.1	The system shall constrain the travel time for the travel route to the specified minimum value if the travel route's travel time is computed to be lower than the specified minimum value.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TravelRouteImpl:computeTravelTime
5.4.1.10	The system shall allow the user to disable travel times for the travel route if travel times are currently enabled for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.10.1	The system shall cease to display the travel time from the travel route on any DMS where it is currently displayed if travel times for the travel route are disabled.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TravelRouteImpl:computeTravelTime, DMSTravInfoMsgDataSupplier:getData
5.4.1.11	The system shall allow the user to enable travel time alerts and/or notifications to be sent if one or more roadway links are specified for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, TravelRouteFactoryImpl:addRoute

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.1.11.1	The system shall require the user to specify a travel time that when exceeded for the travel route will cause the system to issue an alert, notification, or both for the travel route, depending on whether travel time alerts and/or notifications are enabled for the travel route. (This is referred to as the alert/notification travel time).	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, TravelRouteFactoryImpl:addRou
5.4.1.11.2	The system shall allow the user to enable travel time alerts for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.11.2.1	The system shall require the user to specify the operations center to be alerted when the travel route's travel time exceeds the alert/notification travel time, when travel time alerts are enabled for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, TravelRouteFactoryImpl:addRoute
5.4.1.11.2.2	If travel time alerts are enabled for a travel route, the system shall issue an alert to the specified operations center whenever the travel time transitions from a time below the alert/notification travel time to a time equal to or greater than the alert/notification travel time.	Travel Times / Toll Rates	Compute Route Travel Time, Create Travel Time Alert	TravelRouteFactoryImpl:computeTravelTime
5.4.1.11.3	The system shall allow the user to enable travel time notifications for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.11.3.1	The system shall require the user to specify the notification group to be notified when the travel route's travel time exceeds the alert/notification travel time, when travel time notifications are enabled for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRouteTravelRouteFactoryImpl:a ddRoute
5.4.1.11.3.2	If travel time notifications are enabled for a travel route, the system shall issue a notification to the specified notification group whenever the travel time transitions from a time below the alert/notification travel time to a time equal to or greater than the alert/notification travel time.	Travel Times / Toll Rates	Compute Route Travel Time, Send Travel Time Notification	TravelRouteFactoryImpl:computeTravelTime
5.4.1.12	The system shall allow the user to disable travel time alerts for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.13	The system shall allow the user to disable travel time notifications for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.14	The system shall allow the user to specify the number of roadway links whose current data quality may fall below their configured minimum data quality when computing the travel time for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.1.15	The system shall allow the user to enable toll rates for the travel route if the route has a toll rate source specified.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.16	The system shall allow the user to disable toll rates for the travel route if enabled.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.16.1	The system shall cease to display the toll rate for the travel route on any DMS when toll rates for the travel route are disabled.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TravelRouteImpl:computeTravelTime, DMSTravInfoMsgDataSupplier:getData
5.4.1.17	The system shall allow the user to enable toll rate alerts and/or notifications to be sent if toll rates are enabled for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.17.1	The system shall allow the user to enable toll rate alerts for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.17.1.1	The system shall require the user to specify the operations center to be alerted when the travel route's currently active toll rate is cleared.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.17.1.2	If toll rate alerts are enabled for a travel route, the system shall issue an alert to the specified operations center whenever a currently active toll rate for that travel route is cleared, provided there is a currently active toll rate document available in the system.	Travel Times / Toll Rates	Create Toll Rate Alert	TravelRouteFactoryImpl:updateTollRateData
5.4.1.17.2	The system shall allow the user to enable toll rate notifications for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.17.2.1	The system shall require the user to specify the notification group to be notified when the travel route's currently active toll rate is cleared, when toll rate notifications are enabled for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.17.2.2	If toll rate notifications are enabled for a travel route, the system shall issue a notification to the specified notification group whenever a currently active toll rate for that travel route is cleared, provided there is a currently active toll rate document available in the system.	Travel Times / Toll Rates	Send Toll Rate Notification	TravelRouteFactoryImpl:updateTollRateData
5.4.1.18	The system shall allow the user to disable toll rate alerts for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.1.19	The system shall allow the user to disable toll rate notifications for the travel route.	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.2	The system shall allow a suitably privileged user to view travel routes that exist in the system.	Travel Times / Toll Rates	View Travel Route List	viewTravelRoutes
5.4.2.1	The system shall show data for each travel route currently defined in the system.	Travel Times / Toll Rates	View Travel Route List	viewTravelRoutes
5.4.2.1.2	The system shall show the name of each travel route.	Travel Times / Toll Rates	View Travel Route List	viewTravelRoutes
5.4.2.1.3	The system shall show the length of each travel route if the user has selected to display that column.	Travel Times / Toll Rates	View Travel Route List	viewTravelRoutes
5.4.2.1.4	The system shall show travel time related data if travel time is applicable to the travel route. (Travel time is not applicable if the travel route does not have any roadway links defined, if travel times are disabled for the travel route, or if there is no current travel time for the travel route due to a data feed or data quality issue.)	Travel Times / Toll Rates	View Travel Route List	viewTravelRoutes
5.4.2.1.4.1	The system shall show the current Travel Time of each travel route, rounded to the nearest minute, if the user has selected to display that column.	Travel Times / Toll Rates	View Travel Route List	viewTravelRoutes
5.4.2.1.4.2	The system shall show the current Travel Time trend for each travel route if the user has selected to display that column.	Travel Times / Toll Rates	View Travel Route List, View Travel Time Trend	viewTravelRoutes
5.4.2.1.4.2.1	Travel time trend shall be computed by comparing the average of the latest N recent travel times to the average of the earliest N recent travel times, where N is a configurable system-wide travel time sample size, the number of recent travel times to compare.	Travel Times / Toll Rates	View Travel Time Trend	TravelRouteImpl:computeTravelTime
5.4.2.1.4.2.1.1	The recent travel times shall include all travel time computations for the travel route that have occurred within the last hour.	Travel Times / Toll Rates	View Travel Time Trend	TravelRouteImpl:computeTravelTime
5.4.2.1.4.2.1.2	The system shall retain at most 12 recent travel times for a travel route.	Travel Times / Toll Rates	View Travel Time Trend	TravelRouteImpl:computeTravelTime
5.4.2.1.4.2.1.3	The system shall include a configurable system-wide travel time threshold, specified as a percentage, used when determining the travel time trend.	Travel Times / Toll Rates	View Travel Time Trend, Configure Travel Time And Toll Rate Settings	setTravelTimeMiscSettigns

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.2.1.4.2.1.4	The travel time trend shall be “Up” if the recent history has at least 2N travel times and the average of the N later travel times is X percent greater than the average of the N earlier travel times, where N is the system-wide travel time sample size and X is the system-wide travel time trend threshold.	Travel Times / Toll Rates	View Travel Time Trend	TravelRouteImpl:computeTravelTime
5.4.2.1.4.2.1.5	The travel time trend shall be “Down” if the recent history has at least 2N travel times and the average of the N later travel times is X percent less than the average of the N earlier travel times, where N is the system-wide travel time sample size and X is the system-wide travel time trend threshold.	Travel Times / Toll Rates	View Travel Time Trend	TravelRouteImpl:computeTravelTime
5.4.2.1.4.2.1.6	The travel time trend shall be “Flat” if the recent history has less than 2N travel times OR the average of the N later travel times is within X percent of the average of the N earlier travel times, where N is the system-wide travel time sample size and X is the system-wide travel time trend threshold.	Travel Times / Toll Rates	View Travel Time Trend	TravelRouteImpl:computeTravelTime
5.4.2.1.4.2.2	The system shall allow the user to view the recent travel time history that was used to compute the travel time trend.	Travel Times / Toll Rates	View Travel Time Trend	viewTravelRoutes
5.4.2.1.4.3	The system shall show the current Speed of traffic for each travel route if the user has selected to display that column.	Travel Times / Toll Rates	View Travel Route List	viewTravelRoutes
5.4.2.1.4.3.1	The current travel route speed shall be computed as the travel route length divided by the current travel route time and represented as miles per hour.	Travel Times / Toll Rates	View Travel Route List	TravelRouteImpl:computeTravelTime
5.4.2.1.5	The system shall show the current Toll Rate for each travel route if applicable AND the user has selected to display that column. (Toll rate is not applicable if the travel route does not have a toll rate source specified, if toll rates are disabled for the travel route, or if there is no current toll rate due to a data feed issue).	Travel Times / Toll Rates	View Travel Route List	viewTravelRoutes
5.4.2.1.6	The system shall show the other CHART system objects currently using each travel route if the user has selected to display that column.	Travel Times / Toll Rates	View Travel Route List	viewTravelRoutes

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.2.1.6.1	A CHART DMS system object shall be considered to be using a travel route if the travel route is used in any pre-configured DMS message (even if that message is not active).	Travel Times / Toll Rates	View Travel Route List	viewTravelRoutes
5.4.2.1.7	The system shall show the roadway route(s) on which each travel route is located if the user has selected to display that column.	Travel Times / Toll Rates	View Travel Route List	viewTravelRoutes
5.4.2.1.8	The system shall show the direction(s) of traffic flow of each travel route if the user has selected to display that column.	Travel Times / Toll Rates	View Travel Route List	viewTravelRoutes
5.4.2.1.9	The system shall show the county(ies) in which each travel route is located if the user has selected to display that column.	Travel Times / Toll Rates	View Travel Route List	viewTravelRoutes
5.4.2.2	The system shall allow the user to sort the list of travel routes.	Travel Times / Toll Rates	View Travel Route List, Sort Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.2.1	The system shall allow the user to sort the list of travel routes by travel route name.	Travel Times / Toll Rates	Sort Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.2.2	The system shall allow the user to sort the list of travel routes by travel route length.	Travel Times / Toll Rates	Sort Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.2.3	The system shall allow the user to sort the list of travel routes by travel time.	Travel Times / Toll Rates	Sort Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.2.4	The system shall allow the user to sort the list of travel routes by travel time trend.	Travel Times / Toll Rates	Sort Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.2.5	The system shall allow the user to sort the list of travel routes by current speed.	Travel Times / Toll Rates	Sort Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.2.6	The system shall allow the user to sort the list of travel routes by toll rate.	Travel Times / Toll Rates	Sort Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.2.7	The system shall allow the user to sort the list of travel routes by the name of CHART system objects using the travel route.	Travel Times / Toll Rates	Sort Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.2.8	The system shall allow the user to sort the list of travel routes by the roadway route where the travel route is located.	Travel Times / Toll Rates	Sort Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.2.9	The system shall allow the user to sort the list of travel routes by the direction of travel on the travel route.	Travel Times / Toll Rates	Sort Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.2.2.10	The system shall allow the user to sort the list of travel routes by the county in which a travel route is located.	Travel Times / Toll Rates	Sort Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.3	The system shall allow the user to filter the list of travel routes.	Travel Times / Toll Rates	Filter Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.3.1	The system shall allow the user to filter the list of travel routes by travel time.	Travel Times / Toll Rates	Filter Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.3.1.1	Filtering by travel time shall be confined to selection of hard-coded ranges of travel times.	Travel Times / Toll Rates	Filter Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.3.2	The system shall allow the user to filter the list of travel routes by travel time trend.	Travel Times / Toll Rates	Filter Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.3.3	The system shall allow the user to filter the list of travel routes by current speed.	Travel Times / Toll Rates	Filter Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.3.3.1	Filtering by current speed shall be confined to selection of hard-coded ranges of speeds.	Travel Times / Toll Rates	Filter Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.3.4	The system shall allow the user to filter the list of travel routes by toll rate.	Travel Times / Toll Rates	Filter Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.3.5	The system shall allow the user to filter the list of travel routes by roadway route.	Travel Times / Toll Rates	Filter Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.3.6	The system shall allow the user to filter the list of travel routes by travel direction.	Travel Times / Toll Rates	Filter Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.2.3.7	The system shall allow the user to filter the list of travel routes by county.	Travel Times / Toll Rates	Filter Travel Route List	viewTravelRoutes, TravelRouteDynListSupporter:createDynList
5.4.3	The system shall allow a suitably privileged user to view the details of a travel route.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails, getHistoryBucketTimes, toBucketArray
5.4.3.1	The details displayed shall include the current status data for the travel route, if available. (Availability of travel time related data is dependent upon the travel route being configured to include one or more roadway links, travel times being enabled for the travel route, and data of a sufficient quality being supplied for the roadway links. Availability of toll rate data is dependent on a toll rate source being specified for the travel route, toll rates being enabled for the travel route, and the toll rate data being supplied by the toll rate source.)	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.3.1.1	The current status data displayed for the travel route shall include the current travel time if available.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.1.2	The current status data displayed for the travel route shall include the current trend if available.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.1.3	The current status data displayed for the travel route shall include the current speed if available.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.1.4	The current status data displayed for the travel route shall include the current toll rate if available.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.2	The details displayed shall include data for each roadway link included in the travel route (if any).	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.2.1	The data displayed for a roadway link shall include its current travel time.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.2.2	The data displayed for a roadway link shall include its current travel time trend.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.2.2.1	The current travel time trend for a roadway link shall be computed in the same manner as travel time trend is computed for a travel route, as specified in requirement 5.4.2.1.4.2 and its sub-requirements.	Travel Times / Toll Rates	View Travel Route Details, View Travel Time Trend	TravelRouteImpl:updateLinkData
5.4.3.2.3	The data displayed for a roadway link shall include the current speed for the link in MPH.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.2.3.1	The speed displayed for a roadway link shall be the speed provided by the data source for the roadway link (if any).	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.2.4	The data displayed for a roadway link shall include the recent travel time history in minutes and seconds. (INRIX supplies link travel times to the thousandth of a minute.)	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails, getHistoryBucketTimes, toBucketArray
5.4.3.2.4.1	The recent travel time history shall be displayed in 5 minute increments for the period of 1 hour.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails, getHistoryBucketTimes, toBucketArray
5.4.3.2.4.2	When the travel time for a travel route is computed, it shall be included in the recent history interval that began most recently but has not yet ended.	Travel Times / Toll Rates	View Travel Route Details	TravelRouteImpl:computeTravelTime, toBucketArray

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.3.2.4.3	The quality of each displayed travel time for a roadway link shall include a quality indicator of Low, Medium, or High.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails, getHistoryBucketTimes, toBucketArray, TravelRouteFactoryImpl:updateLinkData
5.4.3.2.4.4	If multiple travel time computations are completed within the same recent history interval, the system shall display only the most recent travel time.	Travel Times / Toll Rates	View Travel Route Details	toBucketArray
5.4.3.2.5	The data displayed for a roadway link shall include configuration data for the roadway link.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.2.5.1	The configuration data displayed for a roadway link shall include the name of the system that provides data for the roadway link.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.2.5.2	The configuration data displayed for a roadway link shall include the ID of the roadway link as specified in the system that provides data for the roadway link.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.2.5.3	The configuration data displayed for a roadway link shall include the name of the roadway link as specified in the system that provides data for the roadway link.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.2.5.4	The configuration data displayed for a roadway link shall include the roadway route name or number where the roadway link is located.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.2.5.5	The configuration data displayed for a roadway link shall include the direction of traffic flow on the roadway link.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.2.5.6	The configuration data displayed for a roadway link shall include the length of the roadway link.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.2.5.7	The configuration data displayed for a roadway link shall include the county in which the roadway link is located.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.2.5.8	The configuration data displayed for a roadway link shall include the distance from the prior link included in the travel route (if any).	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.3	The details displayed shall include data pertaining to the toll rate source and toll rate settings for the travel route (if any).	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.3.1	The data displayed for a toll rate source shall include the recent toll rate history.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails, getHistoryBucketTimes, toBucketArray

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.3.3.1.1	The recent toll rate history shall be displayed in 5 minute increments for the period of 1 hour.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails, getHistoryBucketTimes, toBucketArray
5.4.3.3.1.2	If multiple toll rate updates are received within the same recent history interval, the system shall display only the latest toll rate update.	Travel Times / Toll Rates	View Travel Route Details	toBucketArray
5.4.3.3.2	The data displayed for a toll rate source shall include its configuration data.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.3.2.1	The configuration data displayed for a toll rate source shall include the name of the system that supplies the toll rate data for the travel route.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.3.2.2	The configuration data displayed for a toll rate source shall include the start ID for the source.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.3.2.3	The configuration data displayed for a toll rate source shall include the end ID for the source.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.3.2.4	The configuration data displayed for a toll rate source shall include the description of the source.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.3.3	The toll rate settings displayed for a travel route shall include an indication as to whether or not toll rate alerts are enabled for the travel route.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.3.4	The toll rate settings displayed for a travel route shall include the operations center that is to receive toll rate alerts for the travel route (when enabled).	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.3.5	The toll rate settings displayed for a travel route shall include an indication as to whether or not toll rate notifications are enabled for the travel route.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.3.6	The toll rate settings displayed for a travel route shall include the notification group that is to receive toll rate notifications for the travel route (when enabled).	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.4	The details displayed shall include the travel route's preferred and alternate destination names.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.5	The details displayed shall include the location settings for the travel route.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.5.1	The location settings displayed for the travel route shall include the source of the location settings (derived from links, or entered by the user).	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.3.5.2	The location settings displayed for the travel route shall include the county(ies) in which the travel route is located.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.5.3	The location settings displayed for the travel route shall include the roadway route(s) on which the travel route is located.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.5.4	The location settings displayed for the travel route shall include the direction of travel corresponding to each roadway route on which the travel route is located.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.5.5	The location settings displayed for the travel route shall include the length of the travel route (in miles).	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.6	The details displayed shall include the travel time settings for the travel route if the travel route includes one or more roadway links.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.6.1	The travel time settings displayed for a travel route shall include an indication as to whether or not travel times for the travel route are enabled.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.6.2	The travel time settings displayed for a travel route shall include the maximum travel time allowed for the travel route. (Sanity check number, prevents display of numbers higher than this threshold)	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.6.3	The travel time settings displayed for a travel route shall include the minimum travel time allowed for the travel route. (Sanity check number, constrains displayed travel times to this minimum)	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.6.4	The travel time settings displayed for a travel route shall include the alert/notification travel time. (The travel time that when exceeded causes an alert and/or notification to be issued if enabled).	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.6.5	The travel time settings displayed for a travel route shall include an indication as to whether or not travel time alerts are enabled for the travel route.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.6.6	The travel time settings displayed for a travel route shall include the operations center that is to receive travel time alerts for the travel route (when enabled).	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.6.7	The travel time settings displayed for a travel route shall include an indication as to whether or not travel time notifications are enabled for the travel route.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.3.6.8	The travel time settings displayed for a travel route shall include the notification group that is to receive travel time notifications for the travel route (when enabled).	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.6.9	The travel time settings displayed for a travel route shall include the maximum number of links allowed to be below the link data quality threshold for the route's travel time to be used.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.7	The details displayed shall include the toll rate settings for the travel route.	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.3.7.1	The toll rate settings displayed for the travel route shall include an indication as to whether toll rates are enabled for the travel route (if the travel route has a toll rate source specified)	Travel Times / Toll Rates	View Travel Route Details	viewTravelRouteDetails
5.4.4	The system shall allow the user to view the details for a roadway link that is contained in a travel route.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails
5.4.4.1	The link details shall include the current travel time for the link.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails
5.4.4.2	The link details shall include the time when the status was obtained.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails
5.4.4.3	The link details shall include the travel time trend.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails
5.4.4.4	The link details shall include the current speed for the link.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails
5.4.4.5	The link details shall include the recent travel time history for the link.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails, getHistoryBucketTimes, toBucketArray

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.4.6	The link details shall include the external system name.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails
5.4.4.7	The link details shall include the link name.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails
5.4.4.8	The link details shall include the county or counties the link is in.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails
5.4.4.9	The link details shall include the route type of the roadway route.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails
5.4.4.10	The link details shall include the roadway route number or name.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails
5.4.4.11	The link details shall include the travel direction.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails
5.4.4.12	The link details shall include the length of the link.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails
5.4.4.13	The link details shall include the geographic coordinates of the starting point of the link.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails
5.4.4.14	The link details shall include the geographic coordinates of the ending point of the link.	Travel Times / Toll Rates	View Travel Route Roadway Link Details	viewTravelRouteLinkDetails

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
5.4.5	The system shall allow a suitably privileged user to modify a travel route that exists in the system.	Travel Times / Toll Rates	Edit Travel Route, Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute, TravelRouteImpl:setConfig, elements from 5.4.1 (Add Route) requirements
5.4.5.1	The system shall allow the user to modify all travel route attributes as specified in 5.4.1 (“add a travel route to the system”).	Travel Times / Toll Rates	Set Travel Route Properties	addEditTravelRouteForm, addEditTravelRoute
5.4.6	The system shall allow a suitably privileged user to remove a travel route that exists in the system.	Travel Times / Toll Rates	Remove Travel Route	removeTravelRoute, TravelRouteImpl:remove
5.4.6.1	The system shall require the user to confirm their choice to remove a travel route from the system.	Travel Times / Toll Rates	Remove Travel Route	removeTravelRoute
5.4.6.2	The system shall warn the user if they are attempting to remove a travel route that is known to be used by one or more other CHART objects.	Travel Times / Toll Rates	Remove Travel Route	removeTravelRoute
6	PROVIDE TRAVELER INFORMATION	HEADER	N/A	N/A
6.1	BROADCAST INFORMATION. The system shall provide audible and visual or textual display messages to several types of devices. The content of the message and the trigger to activate the device (where necessary) are initiated (or calculated and dynamically updated in the case of queue length and travel time) by an earlier process.	HEADER	N/A	N/A
6.1.2	The system shall broadcast traveler information via DMSs.	HEADER	N/A	N/A
6.1.2.1	The traveler information available for broadcast via DMSs shall include Travel Time.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm , Chart2DMSImpl:addTravInfoMsg
6.1.2.2	The traveler information available for broadcast via DMSs shall include Toll Rates.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm , Chart2DMSImpl:addTravInfoMsg
6.1.2.3	The system shall allow a user with appropriate rights to manage message templates used to broadcast traveler information via DMSs.	Travel Times / Toll Rates	Configure Traveler Information Message Templates	getDMSTravInfoMsgTemplateList, filterDMSTravInfoMsgTemplateList, sortDMSTravInfoMsgTemplateList, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1	The system shall allow the user to create a new DMS message template.	Travel Times / Toll Rates	Add Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm , MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.2.3.1.1	The system shall require the user to specify the size of the DMS (rows / cols) the template applies to.	Travel Times / Toll Rates	Add Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.2	The system shall allow the user to specify the message content for a maximum of 2 pages of a DMS message.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.3	The system shall allow the user to specify the message content to appear on each row of a DMS message page.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.3.1	The message content of a row may include free form text.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.3.2	The message content of a row may include one or more data fields.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.3.2.1	The system shall support a data field type used to include a travel route's current travel time in a message template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.3.2.2	The system shall support a data field type used to include a travel route's current travel time range in a message template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.3.2.3	The system shall support a data field type used to include a travel route's current toll rate in a message template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.3.2.4	The system shall support a data field type used to include a travel route's destination in a message template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.2.3.1.3.2.4.1	The system shall automatically set the width of a destination field to be the maximum width available between the characters and/or fields that occur to the left of destination field and the characters and/or fields that occur to the right of the destination field.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm
6.1.2.3.1.3.2.4.2	The system shall allow the user to manually set the width of a destination field to a value greater than one and less than or equal to the maximum destination width determined by the system.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm
6.1.2.3.1.3.2.5	The system shall support a data field type used to include the system's toll rate effective time in a message template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.3.2.6	The system shall support a data field type used to include a travel route's length in a message template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.4	The system shall allow the user to specify that each row of the DMS message is to be left, center, or right justified. (Note: justification of a row that contains a maximum-width travel route destination has no effect, as the content will be structured to fill the entire row.)	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.5	The system shall allow a single message template to be configured to contain data from multiple travel routes.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.5.1	The system shall use an index to identify the data source for each field.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.5.1.1	Fields with a common index shall have their data supplied by a common data source. (For example, a travel time data field with index 1 and a destination field with index 1 would get both of those pieces of data from the same travel route)	Travel Times / Toll Rates	Specify Traveler Information Message Template	TemplateRow:formatMulti, DMSTravInfoMsgDataSupplier:getData

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.2.3.1.5.2	The system shall allow message templates to contain data fields for at least 6 different data sources.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.6	The system shall allow the user to specify the formats for each field type included in the message template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.6.1	The system shall allow the user to specify one justification to be used within all destination data fields in the template. (The specified justification will be used when a destination name is smaller than the destination field)	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.6.2	The system shall allow the user to specify one format to be used for all travel time fields in the template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.6.3	The system shall allow the user to specify one format to be used for all travel time range fields in the template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.6.4	The system shall allow the user to specify one format to be used for all toll rate fields in the template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.6.5	The system shall allow the user to specify one format to be used for all toll rate time fields in the template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.6.6	The system shall allow the user to specify one format to be used for all travel route length fields in the template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.7	The system shall allow the user to specify how the system should utilize the template when data is unavailable for one or more data fields.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.2.3.1.7.1	The system shall allow the user to specify that the system should not display a message using the template if any data field has missing data.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.7.2	The system shall allow the user to specify that the system should not display any page in a message using the template if any data field on that page is missing data.	Travel Times / Toll Rates	Specify Traveler Information Message Template	STravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.7.3	The system shall allow the user to specify that the system should not display any row in a message using the template if any data field on the row is missing data. (Note that when a template is used in a DMS traveler information message, the user can choose to use automatic row positioning to help format messages with missing rows.)	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.8	The system shall allow the user to specify the page timing to use when displaying a DMS message using the template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.8.1	The system shall allow the user to specify the time each page should be displayed before blanking the sign in preparation display the next page (page on time).	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.8.2	The system shall allow the user to specify the time the DMS should remain blank after displaying a page prior to displaying the next page (page off time).	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.9	The system shall allow the user to enter a description for the message template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate
6.1.2.3.1.10	The system shall indicate when any row of the message template is specified (or attempted to be specified) to be wider than the target sign width of the template.	Travel Times / Toll Rates	Specify Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.2.3.1.11	The system shall show a graphical representation of a sample message, using dummy data in data fields, that could result from use of the message template as currently defined. (Note: it is not possible to use real data as the template is not associated with actual data feeds at this time.)	Travel Times / Toll Rates	View Traveler Information Message True Display, Format Traveler Information Message, Replace Traveler Information Message Template Tags	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm
6.1.2.3.1.11.1	The system shall automatically update the graphical representation of the sample message as valid changes are made to the template. (Valid changes are those that do not make a row exceed the width of the template.)	Travel Times / Toll Rates	View Traveler Information Message True Display	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm
6.1.2.3.1.12	The system shall allow the user to perform a spelling check on the message template per requirement 1.2.1.3.	Travel Times / Toll Rates	Check Traveler Information Message Template Spelling	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm
6.1.2.3.1.13	The system shall prevent a template from being saved if the template contains any words that exist in the system's banned word dictionary (and flagged as applicable to DMS devices).	Travel Times / Toll Rates	Check Traveler Information Message Template For Banned Words	MessageTemplateFactoryImpl:createDMSTravInfoMsgTemplate, MessageTemplateFactoryImpl:setConfig
6.1.2.3.2	The system shall allow the user to edit an existing DMS message template.	Travel Times / Toll Rates	Edit Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateImpl:setConfig
6.1.2.3.2.1	The system shall allow all actions as specified in requirement 6.1.2.3.1 when editing a message template, except that the sign size cannot be changed.	Travel Times / Toll Rates	Edit Traveler Information Message Template	createOrUpdateDMSTravInfoMsgTemplate, submitDMSTravInfoMsgTemplateForm, MessageTemplateImpl:setConfig
6.1.2.3.3	The system shall allow the user to remove an existing DMS message template from the system.	Travel Times / Toll Rates	Remove Traveler Information Message Template	removeDMSTravInfoMsgTemplate, MessageTemplateImpl:remove
6.1.2.3.3.1	The system shall display a warning message asking the user to confirm their action prior to removing a message template.	Travel Times / Toll Rates	Remove Traveler Information Message Template	removeDMSTravInfoMsgTemplate
6.1.2.3.3.2	The system shall prevent a message template from being removed from the system if it is known to be used by any DMS.	Travel Times / Toll Rates	Remove Traveler Information Message Template	removeDMSTravInfoMsgTemplate
6.1.2.3.4	The system shall allow the user to view a list of DMS message templates that exist in the system.	Travel Times / Toll Rates	View Traveler Information Message Templates	getDMSTravInfoMsgTemplateList

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.2.3.4.1	The system shall display data for each message template.	Travel Times / Toll Rates	View Traveler Information Message Templates	getDMSTravInfoMsgTemplateList
6.1.2.3.4.1.1	The system shall display the template name.	Travel Times / Toll Rates	View Traveler Information Message Templates	getDMSTravInfoMsgTemplateList
6.1.2.3.4.1.2	The system shall display the target sign size.	Travel Times / Toll Rates	View Traveler Information Message Templates	getDMSTravInfoMsgTemplateList
6.1.2.3.4.1.3	The system shall display a text representation of the message template, including all fields and text included in the template.	Travel Times / Toll Rates	View Traveler Information Message Templates	getDMSTravInfoMsgTemplateList
6.1.2.3.4.1.4	The system shall display the format specified for travel time fields that exist in the template (if any).	Travel Times / Toll Rates	View Traveler Information Message Templates	getDMSTravInfoMsgTemplateList
6.1.2.3.4.1.5	The system shall display the format specified for travel time range fields that exist in the template (if any).	Travel Times / Toll Rates	View Traveler Information Message Templates	getDMSTravInfoMsgTemplateList
6.1.2.3.4.1.6	The system shall display the format specified for toll rate fields that exist in the template (if any).	Travel Times / Toll Rates	View Traveler Information Message Templates	getDMSTravInfoMsgTemplateList
6.1.2.3.4.1.7	The system shall display the format specified for toll rate time fields that exist in the template (if any).	Travel Times / Toll Rates	View Traveler Information Message Templates	getDMSTravInfoMsgTemplateList
6.1.2.3.4.1.8	The system shall display the format specified for route length fields that exist in the template (if any).	Travel Times / Toll Rates	View Traveler Information Message Templates	getDMSTravInfoMsgTemplateList
6.1.2.3.4.1.9	The system shall display a graphical representation of a sample message that could result from the use of the message template, using dummy data in data fields.	Travel Times / Toll Rates	View Traveler Information Message Templates, View Traveler Information Message True Display	getDMSTravInfoMsgTemplateList
6.1.2.3.4.2	The system shall allow the list of message templates to be sorted.	Travel Times / Toll Rates	View Traveler Information Message Templates	sortDMSTravInfoMsgTemplateList

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.2.3.4.2.1	The system shall allow the list of message templates to be sorted by template name.	Travel Times / Toll Rates	View Traveler Information Message Templates	sortDMSTravInfoMsgTemplateList
6.1.2.3.4.2.2	The system shall allow the list of message templates to be sorted by target sign size.	Travel Times / Toll Rates	View Traveler Information Message Templates	sortDMSTravInfoMsgTemplateList
6.1.2.3.4.3	The system shall allow the list of message templates to be filtered.	Travel Times / Toll Rates	View Traveler Information Message Templates	filterDMSTravInfoMsgTemplateList
6.1.2.3.4.3.1	The system shall allow the list of message templates to be filtered by target sign size.	Travel Times / Toll Rates	View Traveler Information Message Templates	filterDMSTravInfoMsgTemplateList
6.1.2.3.4.3.2	The system shall allow the list of message templates to be filtered by whether or not the template contains any travel time fields.	Travel Times / Toll Rates	View Traveler Information Message Templates	filterDMSTravInfoMsgTemplateList
6.1.2.3.4.3.2.1	The system shall allow the list of message templates to be filtered on any specific travel time field value.	Travel Times / Toll Rates	View Traveler Information Message Templates	filterDMSTravInfoMsgTemplateList
6.1.2.3.4.3.3	The system shall allow the list of message templates to be filtered by whether or not the template contains any toll rate fields.	Travel Times / Toll Rates	View Traveler Information Message Templates	filterDMSTravInfoMsgTemplateList
6.1.2.3.4.3.3.1	The system shall allow the list of message templates to be filtered on any specific toll rate field value.	Travel Times / Toll Rates	View Traveler Information Message Templates	filterDMSTravInfoMsgTemplateList
6.1.2.4	The system shall allow a user with appropriate rights to manage travel routes associated with a DMS.	Travel Times / Toll Rates	Specify DMS Traveler Information Message Settings	viewEditDMSTravelRoutesForm, setDMSTravelRoutes
6.1.2.4.1	The system shall allow the user to view the travel routes associated with a DMS.	Travel Times / Toll Rates	View Associated Travel Routes	GUIDMSDataClasses
6.1.2.4.2	The system shall allow the user to associate travel routes with a DMS.	Travel Times / Toll Rates	Add Travel Route To DMS	viewEditDMSTravelRoutesForm, setDMSTravelRoutes, Chart2DMSImpl:setRelatedRoutes
6.1.2.4.3	The system shall allow the user to disassociate travel routes from a DMS.	Travel Times / Toll Rates	Remove Travel Route From DMS	viewEditDMSTravelRoutesForm, setDMSTravelRoutes, Chart2DMSImpl:setRelatedRoutes

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.2.5	The system shall allow a user with appropriate rights to manage traveler information messages for a DMS.	Travel Times / Toll Rates	Specify DMS Traveler Information Message Settings, View Traveler Info Message Settings	getAddEditDMSTravInfoMsgForm, submitDMSTravInfoMsgForm
6.1.2.5.1	The system shall allow the user to create a traveler information message for use on a specific DMS.	Travel Times / Toll Rates	Add Traveler Information Message	getAddEditDMSTravInfoMsgForm, submitDMSTravInfoMsgForm, Chart2DMSImpl:addTravInfoMsg
6.1.2.5.1.1	The system shall require the user to select a previously defined message template to be used for the traveler information message.	Travel Times / Toll Rates	Specify Traveler Information Message Format	getDMSTravInfoMsgTemplateDataJSON
6.1.2.5.1.1.1	The system shall allow only message templates whose target sign size exactly matches the DMS size (rows and columns) to be selected for use in a traveler information message for the DMS.	Travel Times / Toll Rates	Specify Traveler Information Message Format	getDMSTravInfoMsgTemplateDataJSON, Chart2DMSImpl:addTravInfoMsg
6.1.2.5.1.1.2	The system shall show a text representation of the selected message template.	Travel Times / Toll Rates	Specify Traveler Information Message Format	getAddEditDMSTravInfoMsgForm, submitDMSTravInfoMsgForm
6.1.2.5.1.1.2.1	The text representation of the message template shall identify fields that are to be associated with a common data source. (In other words, if there are 2 fields for one data source and 2 fields for another, the system needs to identify these associations to the user)	Travel Times / Toll Rates	Specify Traveler Information Message Format	getAddEditDMSTravInfoMsgForm, submitDMSTravInfoMsgForm
6.1.2.5.1.2	The system shall require the user to select one or more travel routes that are associated with the DMS for use as data source(s) for the traveler information message.	Travel Times / Toll Rates	Specify Traveler Information Message Format	getAddEditDMSTravInfoMsgForm, submitDMSTravInfoMsgForm
6.1.2.5.1.2.1	The system shall allow the user to select a travel route for each separate data source specified in the message template. (For example, if the message template contains fields with 2 different index numbers, the system shall allow 2 travel route selections)	Travel Times / Toll Rates	Specify Traveler Information Message Format	getAddEditDMSTravInfoMsgForm, submitDMSTravInfoMsgForm
6.1.2.5.1.2.1.1	The system shall allow the user to select “None” as the travel route for a data source specified in the message template.	Travel Times / Toll Rates	Specify Traveler Information Message Format	getAddEditDMSTravInfoMsgForm, submitDMSTravInfoMsgForm

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.2.5.1.2.1.1.1	The system shall handle messages that do not have a travel route selected for a data source specified in the message template using the setting for the template that determines how missing data is to be handled. (See requirement 6.1.2.3.1.7)	Travel Times / Toll Rates	Specify Traveler Information Message Format, Replace Traveler Information Message Template Tags	DMSTravInfoMsgTemplateModel:formatMulti
6.1.2.5.1.2.1.2	The lists of travel routes presented for selection as a data source for the template shall include only those data sources that can provide ALL data fields specified in the template for that source. (For example, if a template has toll rate and travel time fields for a single data source, only travel routes that can provide those 2 pieces of data will be listed).	Travel Times / Toll Rates	Specify Traveler Information Message Format	getAddEditDMSTravInfoMsgForm, submitDMSTravInfoMsgForm
6.1.2.5.1.3	The system shall show a graphical representation of the DMS message that results from the selected message template and data source(s).	Travel Times / Toll Rates	Specify Traveler Information Message Format	DMSTravInfoMsgTrueDisplayMgr:updateGIF, discoverDMSClasses, WebChart2DMS:create, WebChart2DMS:setupDMSTravInfoMsgs, WebChart2DMS:updateConfig, WebChart2DMS:update_ModelChange, WebDMSFactory:createDMS, DynImageCleanupTask:run
6.1.2.5.1.3.1	The system shall use actual data from each data source specified in the message in each data field, when available.	Travel Times / Toll Rates	Specify Traveler Information Message Format, Replace Traveler Information Message Template Tags	WebChart2DMS:update_ModelChange
6.1.2.5.1.3.2	The system shall use dummy data in each data field for which actual data from the source is not available.	Travel Times / Toll Rates	Specify Traveler Information Message Format, Replace Traveler Information Message Template Tags	DMSTravInfoMsgTrueDisplayMgr:updateGIF
6.1.2.5.1.4	The system shall allow the user to specify that the message is to use automatic row positioning.	Travel Times / Toll Rates	Specify Traveler Information Message Format	getAddEditDMSTravInfoMsgForm, submitDMSTravInfoMsgForm
6.1.2.5.1.4.1	When automatic row positioning is enabled for a traveler information message rows of the message shall be automatically positioned..	Travel Times / Toll Rates	Format Traveler Information Message	eDisplayMgr:updateGIF, DMSTravInfoMsgTemplateModel:formatMulti

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.2.5.1.4.1.1	Any phase (page on a sign display) with one line of text shall be displayed on line two on a 3-line or 4-line DMS.	Travel Times / Toll Rates	Format Traveler Information Message	DMSTravInfoMsgTrueDisplayMgr:updateGIF, DMSTravInfoMsgTemplateModel:formatMulti
6.1.2.5.1.4.1.2	Any phase (page on a sign display) with two lines of text shall have that text displayed on lines one and three on a 3-line or 4-line DMS.	Travel Times / Toll Rates	Format Traveler Information Message	DMSTravInfoMsgTrueDisplayMgr:updateGIF, DMSTravInfoMsgTemplateModel:formatMulti
6.1.2.5.1.4.1.3	Any phase (page on a sign display) with three lines of text shall have that text displayed on lines one, two and three on a 3-line or 4-line DMS.	Travel Times / Toll Rates	Format Traveler Information Message	DMSTravInfoMsgTrueDisplayMgr:updateGIF, DMSTravInfoMsgTemplateModel:formatMulti
6.1.2.5.1.5	When a new traveler information message is created for a DMS, the system will by default initialize the message to an inactive state.	Travel Times / Toll Rates	Add Traveler Information Message	submitDMSTravInfoMsgForm, Chart2DMSImpl:addTravInfoMsg
6.1.2.5.2	The system shall allow the user to edit a traveler information message that has been previously configured for a DMS.	Travel Times / Toll Rates	Edit Traveler Information Message	getAddEditDMSTravInfoMsgForm, submitDMSTravInfoMsgForm, Chart2DMSImpl:modifyDMSTravInfoMsg
6.1.2.5.2.1	The system shall allow the user to edit all aspects of the message as detailed in requirement 6.1.2.5.1.	Travel Times / Toll Rates	Edit Traveler Information Message, Specify Traveler Information Message Format	getAddEditDMSTravInfoMsgForm, submitDMSTravInfoMsgForm, Chart2DMSImpl:modifyDMSTravInfoMsg
6.1.2.5.3	The system shall allow the user to view the traveler information messages currently configured for a DMS.	Travel Times / Toll Rates	View DMS Traveler Information Messages	GUIDMSDataClasses
6.1.2.5.3.1	The system shall display a graphical representation of each traveler information message.	Travel Times / Toll Rates	View DMS Traveler Information Messages	DMSTravInfoMsgTrueDisplayMgr:updateGIF
6.1.2.5.3.1.1	The system shall use actual data from each data source specified in the message in each data field, when available.	Travel Times / Toll Rates	View DMS Traveler Information Messages	DMSTravInfoMsgTrueDisplayMgr:updateGIF
6.1.2.5.3.1.2	The system shall use dummy data in each data field for which actual data from the source is not available.	Travel Times / Toll Rates	View DMS Traveler Information Messages	DMSTravInfoMsgTrueDisplayMgr:updateGIF
6.1.2.5.3.2	The system shall display the current status of each traveler information message, (Enabled or Disabled)	Travel Times / Toll Rates	View DMS Traveler Information Messages	GUIDMSDataClasses
6.1.2.5.4	The system shall allow the user to remove a traveler information message from a DMS.	Travel Times / Toll Rates	Remove Traveler Information Message	removeDMSTravInfoMsg, Chart2DMSImpl:removeDMSTravInfoMsg

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.2.5.4.1	The system shall confirm the user's request to remove a traveler information message from a DMS.	Travel Times / Toll Rates	Remove Traveler Information Message	removeDMSTravInfoMsg
6.1.2.5.4.2	The system shall prevent a traveler information message from being removed from a DMS if it is currently enabled for display on that DMS.	Travel Times / Toll Rates	Remove Traveler Information Message	DMSControlModulePkg:Chart2DMSImpl:removeTravInfoMsg
6.1.2.6	The system shall allow a user with appropriate rights to manage the travel time display schedule for a DMS.	Travel Times / Toll Rates	Set DMS Travel Time Display Schedule	setDMSTravelTimeDisplaySchedule, Chart2DMSImpl:setTravelTimeSchedule
6.1.2.6.1	The system shall allow the user to choose to use the system-wide travel time display schedule for the DMS.	Travel Times / Toll Rates	Set DMS Travel Time Display Schedule	setDMSTravelTimeDisplaySchedule, Chart2DMSImpl:setTravelTimeSchedule
6.1.2.6.2	The system shall allow the user to choose to override the system-wide travel time schedule for the DMS.	Travel Times / Toll Rates	Set DMS Travel Time Display Schedule	setDMSTravelTimeDisplaySchedule, Chart2DMSImpl:setTravelTimeSchedule
6.1.2.6.2.1	The system shall allow the user to specify that the DMS is permitted to display travel time messages during all hours of the day (24/7).	Travel Times / Toll Rates	Set DMS Travel Time Display Schedule	setDMSTravelTimeDisplaySchedule, Chart2DMSImpl:setTravelTimeSchedule
6.1.2.6.2.2	The system shall allow the user to specify time periods during the day when the DMS may display travel time messages.	Travel Times / Toll Rates	Set DMS Travel Time Display Schedule	setDMSTravelTimeDisplaySchedule, Chart2DMSImpl:setTravelTimeSchedule
6.1.7	The system shall allow a device to be activated even if it is not used for an event (e.g., to display travel time).	Travel Times / Toll Rates	Activate Traveler Information Message	setDMSTravInfoMsgEnabledFlag, Chart2DMSImpl:setTravInfoMsgEnabledFlag
6.1.7.1	The system shall allow a user to activate a pre-configured traveler information message on a DMS.	Travel Times / Toll Rates	Enable Traveler Information Message	setDMSTravInfoMsgEnabledFlag, Chart2DMSImpl:setTravInfoMsgEnabledFlag
6.1.7.1.1	When a traveler information message is activated on a DMS, the system shall automatically create a DMS message based on the message template and data sources specified for the message.	Travel Times / Toll Rates	Activate Traveler Information Message, Format Traveler Information Message, Replace Traveler Information Message Tags	Chart2DMSImpl:setTravInfoMsgEnabledFlag, DMSControlModulePkg:/EnabledTravInfoMsgCmd:execute, DMSTravInfoMsgHandler:checkMessage
6.1.7.1.1.1	The system shall replace any destination fields in the message template with a destination name from the travel route specified as the data source for the field.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti, DMSTravInfoMsgDataSupplier:getData

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.7.1.1.1.1	The system shall use the preferred, alternate 1, or alternate 2 destination name specified in a travel route which is the longest destination name that will fit within the width of the destination field as specified in the message template.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti, DMSTravInfoMsgDataSupplier:getData
6.1.7.1.1.1.2	The system shall justify the destination name within the destination field in the message template according to the destination justification setting specified in the message template.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti
6.1.7.1.1.1.3	If the travel route specified as the data source for a destination field is not available, the system shall alter or disable the traveler information message as specified by the missing data setting for the message template. (Remove the row containing the destination field, remove the page containing the destination field, or disable the entire message)	Travel Times / Toll Rates	Format Traveler Information Message	DMSTravInfoMsgTemplateModel:formatMulti
6.1.7.1.1.2	The system shall replace any travel time fields in the message template with the current travel time from the travel route specified as the data source for the field.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti, DMSTravInfoMsgDataSupplier:getData
6.1.7.1.1.2.1	If the travel time field is for the actual travel time, the actual specific travel time shall be used.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti, DMSTravInfoMsgDataSupplier:getData
6.1.7.1.1.2.2	If the travel time field is for a travel time range, the actual travel time shall be mapped into a range according to a system-wide travel time range configuration, and the range shall be displayed instead of the actual travel time.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti, DMSTravInfoMsgDataSupplier:getData
6.1.7.1.1.2.2.1	The system shall allow travel time ranges to be defined by adding / subtracting a specified number of minutes from the actual travel time.	Travel Times / Toll Rates	Configure Travel Time And Toll Rate Settings	getTravelTimeRangesForm, setTravelTimeRanges, TravelTimeRange:constructor
6.1.7.1.1.2.2.2	The system shall allow the number of minutes added / subtracted to obtain a travel time range to be configurable based on the range in which the actual travel time falls. (Example: travel time 1 – 10 minutes, add/subtract 1 minute to obtain the range. Travel time 11 – 20 minutes, add/subtract 2 minutes to obtain the range, etc.)	Travel Times / Toll Rates	Configure Travel Time And Toll Rate Settings	getTravelTimeRangesForm, setTravelTimeRanges, TravelTimeRange:constructor
6.1.7.1.1.2.3	The travel time data shall be displayed according to the format specified within the message template.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.7.1.1.2.4	The travel time data shall be right justified within the travel time field in the message template.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti
6.1.7.1.1.2.5	If the current travel time falls below the minimum travel time specified for the travel route that is specified as the data source for the travel time, the minimum travel time shall be used in place of the actual travel time. (This applies whether the actual travel time is displayed or a travel time range. In the case of a range, the range will be obtained using the minimum travel time)	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TravelRouteImpl:computeTravelTime
6.1.7.1.1.2.6	If the travel time data is unavailable or disabled for the travel route specified as the data source for a travel time field, the system shall alter or disable the traveler information message as specified by the missing data setting for the message template. (Remove the row containing the travel time field, remove the page containing the travel time field, or disable the entire message.)	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	DMSTravInfoMsgTemplateModel:formatMulti
6.1.7.1.1.2.6.1	Travel time data for a travel route shall be considered disabled if the travel route no longer exists in the system.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	DMSTravInfoMsgDataSupplier:getData
6.1.7.1.1.2.6.2	The travel time data for a travel route shall be considered disabled if travel times are disabled for the travel route.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TravelRouteImpl:computeTravelTime, DMSTravInfoMsgDataSupplier:getData
6.1.7.1.1.2.6.3	The travel time data for a travel route shall be considered unavailable if the travel time for the travel route does not meet the minimum quality setting for N or more roadway links in the travel route, where N is a setting specified for the travel route.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TravelRouteImpl:computeTravelTime , DMSTravInfoMsgDataSupplier:getData
6.1.7.1.1.2.6.4	The travel time data for a travel route shall be considered unavailable if the travel time for the travel route cannot be obtained from its underlying source (for example from INRIX).	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TravelRouteImpl:computeTravelTime
6.1.7.1.1.2.6.5	The travel time data for a travel route shall be considered unavailable if the travel time for the travel route exceeds the maximum displayed travel time as specified for the travel route.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TravelRouteImpl:computeTravelTime , DMSTravInfoMsgDataSupplier:getData

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.7.1.1.2.6.6	The travel time data for a travel route shall be considered unavailable if the travel route does not have any roadway links specified.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	StalenessWatcherTimerTask in TravelRouteModule
6.1.7.1.1.3	The system shall replace any toll rate fields in the message template with the current toll rate data from the travel route specified as the data source for the field.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti, DMSTravInfoMsgDataSupplier:getData
6.1.7.1.1.3.1	The system shall format the toll rate according to the toll rate format specified in the message template.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti
6.1.7.1.1.3.2	The system shall right justify the toll rate within the toll rate field in the message template.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti
6.1.7.1.1.3.3	If the toll rate data is unavailable or disabled for the travel route specified as the data source for a toll rate field, the system shall alter or disable the traveler information message as specified by the missing data setting for the message template. (Remove the row containing the toll rate field, remove the page containing the toll rate field, or disable the entire message.)	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	DMSTravInfoMsgTemplateModel:formatMulti
6.1.7.1.1.3.3.1	The toll rate data for a travel route shall be considered unavailable if the travel route is no longer defined in the system.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	DMSTravInfoMsgDataSupplier:getData
6.1.7.1.1.3.3.2	The toll rate data for a travel route shall be considered disabled if toll rates are disabled for the travel route.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TravelRouteFactoryImpl:updateTollData
6.1.7.1.1.3.3.3	The toll rate data for a travel route shall be considered unavailable if the toll rate data is not present for the travel route. (For example, if Vector has not yet provided data for the travel route).	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TravelRouteFactoryImpl:updateTollData
6.1.7.1.1.4	The system shall replace any toll rate time fields in the message template with the latest toll rate time from any toll rate source specified for inclusion in the message. (Each toll rate may have its own toll rate time, but in practice only one time will be included in a message, and the times for all toll rates will match, so the latest time for any toll rate in the message will be used)	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	DMSTravInfoMsgDataSupplier:getData

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.7.1.1.4.1	The system shall format the toll rate time according to the toll rate time format specified in the message template.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti
6.1.7.1.1.4.2	The system shall right justify the toll rate time within the toll rate time field in the message template.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti
6.1.7.1.1.4.3	If the toll rate time data is unavailable or disabled, the system shall alter or disable the traveler information message as specified by the missing data setting for the message template. (Remove the row containing the toll rate time field, remove the page containing the toll rate time field, or disable the entire message.)	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	DMSTravInfoMsgTemplateModel:formatMulti
6.1.7.1.1.4.3.1	The toll rate time data shall be considered unavailable if all travel routes specified as the data sources for toll rate fields within the message are no longer defined in the system.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	DMSTravInfoMsgDataSupplier:getData
6.1.7.1.1.4.3.2	The toll rate time data shall be considered disabled if toll rates are disabled for all travel routes specified as sources for toll rate data fields within the message.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	DMSTravInfoMsgDataSupplier:getData
6.1.7.1.1.4.3.3	The toll rate time data shall be considered unavailable if toll rate time data is not available for any of the travel routes specified as sources for toll rate data fields within the message.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	DMSTravInfoMsgDataSupplier:getData
6.1.7.1.1.4.3.4	The toll rate time data shall be considered disabled if there are no toll rate data sources specified for the message, or all toll rate time fields are disabled/unavailable. (This includes the cases where there are no toll rate fields specified in the message template, there are no travel routes assigned for any of those fields, or data is not available is disabled for all of those fields. In other words, toll rate times will not be included in the message if toll rates are not)	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	DMSTravInfoMsgDataSupplier:getData
6.1.7.1.1.5	The system shall replace any distance fields in the message template with the length of the travel route specified as the data source for the field.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	DMSTravInfoMsgDataSupplier:getData, TemplateRow:formatMulti
6.1.7.1.1.5.1	The system shall format the distance according to the distance format specified in the message template.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.7.1.1.5.2	The system shall right justify the distance within the distance field.	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	TemplateRow:formatMulti
6.1.7.1.1.5.3	If the travel route specified as the source for the distance field is unavailable, the system shall alter or disable the traveler information message as specified by the missing data setting for the message template. (Remove the row containing the distance field, remove the page containing the distance field, or disable the entire message.)	Travel Times / Toll Rates	Replace Traveler Information Message Template Tags	DMSTravInfoMsgTemplateModel:formatMulti
6.1.7.1.2	When a traveler information message is activated on a DMS, the DMS shall place the message in its arbitration queue in the appropriate “bucket”.	Travel Times / Toll Rates	Activate Traveler Information Message	DMSTravInfoMsgHandler:checkMessage
6.1.7.1.2.1	The system shall place all DMS traveler information messages that contain one or more toll rate fields into the specified toll rate arbitration queue bucket for the DMS according to 1.5.2.1.4.19. (If a message contains only toll rates, or if it contains toll rates AND travel times, it goes in this toll rate bucket)	Travel Times / Toll Rates	Activate Traveler Information Message	DMSTravInfoMsgHandler:checkMessage
6.1.7.1.2.2	The system shall place all DMS traveler information messages that contain travel time or travel time range fields but NOT any toll rate fields into the specified travel time arbitration queue bucket for the DMS according to 1.5.2.1.4.18. (If a message has travel times AND toll rates, the message goes in the bucket for toll rates as identified in 6.1.7.1.2.1 instead)	Travel Times / Toll Rates	Activate Traveler Information Message	DMSTravInfoMsgHandler:checkMessage
6.1.7.1.2.3	The system shall place all DMS traveler information messages that do not contain travel time, travel time range, or toll rate fields into the specified travel time arbitration queue bucket for the DMS according to 1.5.2.1.4.18.	Travel Times / Toll Rates	Activate Traveler Information Message	DMSTravInfoMsgHandler:checkMessage
6.1.7.1.3	The system shall utilize the DMS message arbitration rules specified in requirement 4.3.1.6 to determine if an active traveler information message is to be displayed on the DMS.	Travel Times / Toll Rates	Activate Traveler Information Message	N/A (existing implementation will work as is)
6.1.7.1.4	The system shall utilize the DMS message combination rules specified in requirement 4.3.1.6 to determine if an active traveler information message is to be combined with another message on the DMS arbitration queue.	Travel Times / Toll Rates	Activate Traveler Information Message	N/A (existing implementation will work as is)

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.7.1.5	The system shall allow only one traveler information message to be active at any time on a single DMS.	Travel Times / Toll Rates	Activate Traveler Information Message	Chart2DMSImpl:setTravInfoMsgEnabledFlag
6.1.7.1.5.1	The system shall deactivate any other traveler information message currently active for a DMS prior to activating a different traveler information message.	Travel Times / Toll Rates	Activate Traveler Information Message	Chart2DMSImpl:setTravInfoMsgEnabledFlag
6.1.7.1.6	The system shall update traveler information messages as new data is provided by the message's data sources.	Travel Times / Toll Rates	Update Traveler Information Message	Chart2DMSImpl:RouteUpdate
6.1.7.1.6.1	The system shall maintain the location of the DMS message within the arbitration queue when a traveler information message is updated.	Travel Times / Toll Rates	Update Traveler Information Message	N/A (existing implementation will work as is)
6.1.7.1.6.2	The system shall create the updated message as specified in 6.1.7.1.1.	Travel Times / Toll Rates	Format Traveler Information Message, Replace Traveler Information Message Tags	Covered by elements specified within all the 6.1.7.1.1 requirements.
6.1.7.2	The system shall allow a user to deactivate a traveler information message that is currently active on a DMS.	Travel Times / Toll Rates	Disable Traveler Information Message	setDMSTravInfoMsgEnabledFlag
6.1.7.2.1	The system shall remove the traveler information message from the DMS arbitration queue when the traveler information message is deactivated.	Travel Times / Toll Rates	Deactivate Traveler Information Message	Chart2DMSImpl:setTravInfoMsgEnabledFlag
6.1.7.3	The system shall automatically remove a traveler information message that is active from the arbitration queue of a DMS when data required to construct the message (as specified in 6.1.7.1.1) is missing and the application of the missing data setting specified in the message template results in the message being disabled or with zero pages.	Travel Times / Toll Rates	Deactivate Traveler Information Message	DMSTravInfoMsgHandler:checkMessage
6.1.7.4	The system shall log all traveler information messages that are displayed on a DMS in the operations log.	Travel Times / Toll Rates	Activate Traveler Information Message, Update Traveler Information Message	N/A (existing implementation will work as is)
6.1.7.5	The system shall log a message in the operations log when a user deactivates a traveler information message.	Travel Times / Toll Rates	Deactivate Traveler Information Message	Chart2DMSImpl:setTravInfoMsgEnabledFlag

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
6.1.7.6	The system shall log a message in the operations log when the system automatically removes a traveler information message from a DMS arbitration queue due to missing data.	Travel Times / Toll Rates	Deactivate Traveler Information Message	DMSTravInfoMsgHandler:checkMessage
6.1.7.7	The system shall activate a traveler information message according to a DMS's travel time schedule, if the message contains no toll rate tags and is currently enabled.	Travel Times	Monitor Travel Time Schedule	Chart2DMSFactoryImpl:checkTravInfoMsgSchedule
6.1.7.8	The system shall deactivate a traveler information message according to a DMS's travel time schedule, if the message contains no toll rate tags and is currently active.	Travel Times	Monitor Travel Time Schedule	Chart2DMSFactoryImpl:checkTravInfoMsgSchedule
9	SYSTEM MAINTAINABILITY, AVAILABILITY, SECURITY, AND DATA DISTRIBUTION	HEADER	N/A	
9.4	Data Distribution	HEADER	N/A	
9.4.1	The system shall allow a system administrator to control a user's access to information in the system by granting/denying user functional rights.	Public / Private Data Sharing	Set Role Functional Rights, Set User Roles	addEditRole
9.4.1.1	The system shall allow a system administrator to control a user's access to Traffic Event related information.	Public / Private Data Sharing	Set Traffic Event Rights	addEditRole
9.4.1.1.2	The system shall allow a system administrator to control a user's access to External Traffic Events by granting/denying the View External Traffic Events functional right.	Public / Private Data Sharing	Set Traffic Event Rights	addEditRole
9.4.1.1.3	The system shall allow a system administrator to control a user's access to view sensitive incident details associated with Traffic Events by granting/denying the View Traffic Event Sensitive Incident Details functional right.	Public / Private Data Sharing	Set Traffic Event Rights	addEditRole
9.4.1.1.4	The system shall allow a system administrator to control a user's access to Traffic Event's log entries by granting/denying the View Traffic Event Log functional right on a per organization basis.	Public / Private Data Sharing	Set Traffic Event Rights	addEditRole
9.4.1.2	The system shall allow a system administrator to control a user's access to device related information.	Public / Private Data Sharing	Set Role Functional Rights, Set User Roles	addEditRole
9.4.1.2.1	The system shall allow a system administrator to control a user's access to External DMS devices by granting/denying the View External DMS functional right.	Public / Private Data Sharing	Set DMS Rights, View DMS List	addEditRole

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
9.4.1.2.2	The system shall allow a system administrator to control a user's access to External TSS devices by granting/denying the View External TSS functional right.	Public / Private Data Sharing	Set TSS Rights, View Detector List	addEditRole
9.4.1.2.3	The system shall allow a system administrator to control a user's access to sensitive configuration data associated with DMS devices by granting/denying the View DMS Sensitive Config functional right on a per organization basis.	Public / Private Data Sharing	Set DMS Rights	addEditRole
9.4.1.2.4	The system shall allow a system administrator to control a user's access to sensitive configuration data associated with TSS devices by granting/denying the View TSS Sensitive Config functional right on a per organization basis.	Public / Private Data Sharing	Set TSS Rights, View Detector Details	addEditRole
9.4.1.2.5	The system shall allow a system administrator to control a user's access to sensitive configuration data associated with HAR and SHAZAM devices by granting/denying the View HAR Sensitive Config functional right on a per organization basis.	Public / Private Data Sharing	Set HAR and SHAZAM Rights	addEditRole
9.4.1.2.6	The system shall allow a system administrator to control a user's access to sensitive configuration data associated with Cameras by granting/denying the View Camera Sensitive Config functional right on a per organization basis.	Public / Private Data Sharing	Set Camera Rights	addEditRole
9.4.1.2.7	The system shall allow a system administrator to control a user's access to sensitive configuration data associated with Monitors by granting/denying the View Monitor Sensitive Config functional right on a per organization basis.	Public / Private Data Sharing	Set Monitor Rights	addEditRole
9.4.1.2.8	The system shall allow a system administrator to control a user's access to TSS detailed traffic parameters (VSO data) by granting/denying the View VSO Detailed Data functional right on a per organization basis.	Public / Private Data Sharing	Set TSS Rights, View Detector List	addEditRole
9.4.1.2.9	The system shall allow a system administrator to control a user's access to TSS summarized traffic parameters (VSO data) by granting/denying the View VSO Summary Data functional right on a per organization basis	Public / Private Data Sharing	Set TSS Rights, View Detector List	addEditRole
9.4.2	The system shall protect certain information maintained in the system by limiting a user's access to / view of that information based on the user's functional rights.	Public / Private Data Sharing	Set Role Functional Rights, Set User Roles	addEditRole
9.4.2.1	The system shall allow a suitably privileged user to view all external Traffic Events currently in the system (View External Traffic Event functional right).	Public / Private Data Sharing	Set Traffic Event Rights	getOpenTrafficEventsXML

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
9.4.2.2	The system shall allow a suitably privileged user to view sensitive information associated with Traffic Events of type incident which convey details about fatalities (View Traffic Event Sensitive Incident Details functional right). This information is conveyed through the incident type and event name.	Public / Private Data Sharing	Set Traffic Event Rights	viewEventDetails
9.4.2.3	The system shall allow a suitably privileged user to view Traffic Event Log entries (View Traffic Event Log functional right).	Public / Private Data Sharing	Set Traffic Event Rights	viewEventDetails
9.4.2.4	The system shall allow a suitably privileged user to view all external DMS devices currently in the system (View External DMS functional right).	Public / Private Data Sharing	Set DMS Rights, View DMS List	chartlite.servlet.dms_dynlist_classes, DMSControlClassDiagram-ExternalDMS CD, ExternalDMS:getStatus SD, ExternalDMS:updateStatus SD, ExternalDMS:getConfiguration SD,
9.4.2.5	The system shall allow a suitably privileged user to view all external TSS devices currently in the system (View External TSS functional right).	Public / Private Data Sharing	Set TSS Rights, View TSS List	chartlite.servlet.tss_dynlist_classes
9.4.2.6	The system shall allow a suitably privileged user to view sensitive configuration data associated with a DMS (View DMS Sensitive Config functional right).	Public / Private Data Sharing	Set DMS Rights	GUIDMSDataClasses
9.4.2.6.1	Sensitive configuration data for a DMS includes: Comm Settings including: Drop Address, Port Mgr. Connection Timeout, Port Type, Baud, Data Bits, Parity, Stop Bits, Flow Control, Def Phone Number (ISDN & POTS), per Port Manager phone numbers.	Public / Private Data Sharing	Set DMS Rights	GUIDMSDataClasses
9.4.2.7	The system shall allow a suitably privileged user to view sensitive configuration data associated with a TSS (View TSS Sensitive Config functional right).	Public / Private Data Sharing	Set TSS Rights, View Detector Details	GUITSSDataClasses
9.4.2.7.1	Sensitive configuration data for a TSS includes: Comm Settings including: Comm Settings including: Drop Address, Port Mgr, Port Type, Baud, Data Bits, Parity, Stop Bits, Flow Control, Def Phone Number (ISDN & POTS), per Port Manager phone numbers.	Public / Private Data Sharing	Set TSS Rights, View Detector Details	GUITSSDataClasses
9.4.2.8	The system shall allow a suitably privileged user to view sensitive configuration data associated with a HAR (View HAR Sensitive Config functional right).	Public / Private Data Sharing	Set HAR and SHAZAM Rights	GUIHARDataClasses

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
9.4.2.8.1	Sensitive configuration data for a HAR includes: Default Phone Number, Access Code, Port Manager Connection Timeout, Port Type, per Port Manager phone numbers.	Public / Private Data Sharing	Set HAR and SHAZAM Rights	GUIHARDataClasses
9.4.2.9	The system shall allow a suitably privileged user to view sensitive configuration data associated with a SHAZAM (View HAR Sensitive Config functional right)	Public / Private Data Sharing	Set HAR and SHAZAM Rights	GUISHAZAMClasses
9.4.2.9.1	Sensitive configuration data for a SHAZAM includes: Default Phone Number, Access Code, Port Manager Connection Timeout, Port Type, per Port Manager phone numbers.	Public / Private Data Sharing	Set HAR and SHAZAM Rights	GUISHAZAMClasses
9.4.2.10	The system shall allow a suitably privileged user to view sensitive configuration data associated with a Camera (View Camera Sensitive Config functional right).	Public / Private Data Sharing	Set Camera Rights	GUIVideoDataClasses
9.4.2.10.1	Sensitive configuration data for a Camera with a Sending Device of type Encoder includes: IP Video Fabric, IP, Encoder Type, Port, Multicast Address, Multicast port.	Public / Private Data Sharing	Set Camera Rights	GUIVideoDataClasses
9.4.2.10.2	Sensitive configuration data for a Camera with a Sending Device of type Switch includes: Switch, Input Port.	Public / Private Data Sharing	Set Camera Rights	GUIVideoDataClasses
9.4.2.10.2	Sensitive configuration data for a Camera with a Control Device of type IP Via Codec includes: IP, Port, Baud, Bits, Parity, Stop Bits, Flow Control.	Public / Private Data Sharing	Set Camera Rights	GUIVideoDataClasses
9.4.2.10.3	Sensitive configuration data for a Camera with a Control Device of type Comm Port includes: Port, Baud, Bits, Parity, Stop Bits, Flow Control.	Public / Private Data Sharing	Set Camera Rights	GUIVideoDataClasses
9.4.2.10.4	Sensitive configuration data for a Camera with a Control Device of type Command Processor includes: Command Processor name.	Public / Private Data Sharing	Set Camera Rights	GUIVideoDataClasses
9.4.2.11	The system shall allow a suitably privileged user to view sensitive configuration data associated with a Monitor (View Monitor Sensitive Config functional right).	Public / Private Data Sharing	Set Monitor Rights	GUIVideoDataClasses
9.4.2.11.1	Sensitive configuration data for a Monitor with a Receiving Device of type Decoder includes IP Video Fabric, IP, Type, TCP Port Video Port.	Public / Private Data Sharing	Set Monitor Rights	GUIVideoDataClasses
9.4.2.11.2	Sensitive configuration data for a Monitor with a Receiving Device of type Switch includes Switch, Output Port.	Public / Private Data Sharing	Set Monitor Rights	GUIVideoDataClasses
10	SYSTEM INTEGRATION	HEADER	N/A	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.1	The system shall interface with other regional ATMS's in the area. Suggestion/example to be validated: RITIS, Regional 911, IEN, CAPWIN, EMMA/MEGIN, WEBEOC, 511, etc.	EXISTING & FUTURE	N/A	
10.1.1	The system shall support the SAE ATIS J2354 standard for event data exchange with external systems (e.g. RITIS).	EXISTING & FUTURE	N/A	
10.1.1.1	The system shall support the importation of event data from external systems using the SAE ATIS J2354 standard.	EXISTING	Import RITIS Event Data	EventAtisImportChartClasses, handleExternalImport
10.1.1.1.1	The system shall translate SAE ATIS J2354 standard formatted event data from RITIS into a compatible CHART external event.	EXISTING	Translate RITIS Event Data	EventAtisImportTranslationClasses, handleEITranslationTask, eventTranlsationStep1Translate
10.1.1.2	The system shall export event data to external systems using the SAE ATIS J2354 standard.	External Interface	Provide Traffic Event Data To External Systems	TrafficEventExportHandler:getTrafficEventList
10.1.1.2.1	The system shall translate CHART event data into SAE ATIS J2354 standard formatted event data with CHART extensions.	External Interface	Provide Traffic Event Data To External Systems	TrafficEventExportHandler:getTrafficEventList
10.1.1.2.2	The system shall support a method for external systems to obtain an inventory of CHART events.	External Interface	Provide Traffic Event Data To External Systems	TrafficEventRequesthandler:processRequest, TrafficEventExportHandler:getTrafficEventList
10.1.1.2.3	The system shall support a method for external system to receive updates to the status of CHART events	External Interface	Provide Traffic Event Data To External Systems	TrafficEventRequesthandler:processRequest, TrafficEventExportHandler:getTrafficEventList
10.1.3	The system shall support the TMDD standard for DMS data exchange with external systems.	External Interface	Provide DMS Data to External Systems, Import RITIS DMS Data	(see sub reqs for details)
10.1.3.1	The system shall support the import of DMS data from external systems using the TMDD standard.	External Interface	Import RITIS DMS Data	DMSImportTranslationClasses, handleDITranslationTask, dmsTranslationStep1Translate
10.1.3.1.1	The system shall translate DMS data in TMDD standard format with RITIS extensions into compatible CHART external DMS formatted data.	External Interface	Translate RITIS DMS Data, Create CHART External DMS	DMSImportTranslationClasses, handleDITranslationTask, dmsTranslationStep1Translate
10.1.3.2	The system shall support the export of DMS data to external systems using the TMDD standard.	External Interface	Provide DMS Data To External Systems	See sub reqs.

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.1.3.2.1	The system shall translate CHART DMS formatted data into TMDD standard formatted DMS data with CHART extensions.	External Interface	Provide DMS Data To External Systems	DMSRequestHandlerProcessRequest, getDMSInventoryList, getDMSStatusList
10.1.3.2.2	The system shall support a method for external systems to obtain an inventory of CHART DMSs.	External Interface	Provide DMS Data To External Systems	DMSRequestHandlerProcessRequest, getDMSInventoryList,
10.1.3.2.3	The system shall support a method for external system to receive updates to the CHART DMS inventory.	External Interface	Provide DMS Data To External Systems	DMSRequestHandlerProcessRequest, getDMSInventoryList,
10.1.3.2.4	The system shall support a method for external systems to obtain the status of CHART DMSs.	External Interface	Provide DMS Data To External Systems	DMSRequestHandlerProcessRequest, getDMSStatusList,
10.1.3.2.5	The system shall support a method for external system to receive updates to the status of CHART DMSs.	External Interface	Provide DMS Data To External Systems	DMSRequestHandlerProcessRequest, getDMSStatusList,
10.1.4	The system shall support the TMDD standard for TSS data exchange with external systems	External Interface	Provide Detector Data To External Systems, Import RITIS Detector Data	See sub reqs.
10.1.4.1	The system shall support the import of TSS data from external systems using the TMDD standard.	External Interface	Import RITIS Detector Data	TSSImportTranslationClasses, handleTITranslationTask, tssTranslationStep1Translate
10.1.4.1.1	The system shall translate TSS data in TMDD standard format with RITIS extensions into compatible CHART external TSS formatted data.	External Interface	Translate RITIS Detector Data, Create External TSS	handleTITranslationTask, tssTranslationStep1Translate
10.1.4.2	The system shall support the export of TSS data to external systems using the TMDD standard.	External Interface	Provide Detector Data to External Systems	Use case only
10.1.4.2.1	The system shall translate CHART TSS formatted data into TMDD standard formatted TSS data with CHART extensions.	External Interface	Provide Detector Data to External Systems	Use case only
10.1.4.2.2	The system shall support a method for external systems to obtain an inventory of CHART TSSs.	External Interface	Provide Detector Data to External Systems	Use case only
10.1.4.2.3	The system shall support a method for external system to receive updates to the CHART TSS inventory.	External Interface	Provide Detector Data to External Systems	Use case only
10.1.4.2.4	The system shall support a method for external systems to obtain the status of CHART TSSs.	External Interface	Provide Detector Data to External Systems	Use case only

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.1.4.2.5	The system shall support a method for external system to receive updates to the status of CHART TSSs.	External Interface	Provide Detector Data to External Systems	Use case only
10.1.5	The system shall support the TMDD standard for HAR data exchange with external systems.	External Interface	Provide HAR Data to External Systems	Use case only
10.1.5.1	The system shall support the import of HAR data from external systems using the TMDD standard. [FUTURE, NEW]	FUTURE	N/A	
10.1.5.1.1	The system shall translate HAR data in TMDD standard format with RITIS extensions into compatible CHART external TSS formatted data. [FUTURE, NEW]	FUTURE	N/A	
10.1.5.2	The system shall support the export of HAR data to external systems using the TMDD standard.	External Interface	Provide HAR Data To External Systems	Use case only
10.1.5.2.1	The system shall translate CHART HAR formatted data into TMDD standard formatted HAR data with CHART extensions.	External Interface	Provide HAR Data To External Systems	Use case only
10.1.5.2.2	The system shall support a method for external systems to obtain an inventory of CHART HAR.	External Interface	Provide HAR Data To External Systems	Use case only
10.1.5.2.3	The system shall support a method for external system to receive updates to the CHART HAR inventory.	External Interface	Provide HAR Data To External Systems	Use case only
10.1.5.2.4	The system shall support a method for external systems to obtain the status of CHART HARs.	External Interface	Provide HAR Data To External Systems	Use case only
10.1.5.2.5	The system shall support a method for external system to receive updates to the status of CHART HARs.	External Interface	Provide HAR Data To External Systems	Use case only
10.1.6	The system shall support the TMDD standard for beacon (SHAZAM) data exchange with external systems.	External Interface	Provide SHAZAM Data To External Systems	Use case only
10.1.6.1	The system shall support the import of beacon data from external systems using the TMDD standard. [FUTURE, NEW]	FUTURE	N/A	
10.1.6.1.1	The system shall translate beacon data in TMDD standard format with RITIS extensions into compatible CHART external beacon formatted data. [FUTURE, NEW]	FUTURE	N/A	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.1.6.2	The system shall support the export of beacon data to external systems using the TMDD standard.	External Interface	Provide SHAZAM Data To External Systems	Use case only
10.1.6.2.1	The system shall translate CHART beacon formatted data into TMDD standard formatted beacon data with CHART extensions.	External Interface	Provide SHAZAM Data To External Systems	Use case only
10.1.6.2.2	The system shall support a method for external systems to obtain an inventory of CHART beacons.	External Interface	Provide SHAZAM Data To External Systems	Use case only
10.1.6.2.3	The system shall support a method for external system to receive updates to the CHART beacon inventory.	External Interface	Provide SHAZAM Data To External Systems	Use case only
10.1.6.2.4	The system shall support a method for external systems to obtain the status of CHART beacons.	External Interface	Provide SHAZAM Data To External Systems	Use case only
10.1.6.2.5	The system shall support a method for external system to receive updates to the status of CHART beacons.	External Interface	Provide SHAZAM Data To External Systems	Use case only
10.7	The system shall support external connections.	EXISTING	N/A	
10.7.1	All external connections shall be made in conjunction with MDOT security policy (e.g., through a protected facility to restrict public access to and limit compromise of the back-end CHART servers).	EXISTING	N/A	
10.7.2	The system shall monitor and maintain the state of external connections established by the system.	EXISTING	Monitor External System Connection	
10.7.2.1	The system shall provide an indication to the users of any connections which are detected to be down.	EXISTING	View External Connection Status, Configure External Connection Alert and Notification Settings	viewExternalConnectionStatus

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.7.2.1.1	If configured to do so, the system shall generate an External Connection Alert when an external connection transitions to the “Failed” state and remains there for more than a configurable connection-specific amount of time.	External Interface	Create External Connection Alert	getExternalConnectionAlertAndNotificationSettingsForm, setExternalConnectionAlertAndNotificationSettingsForm, ExternalSystemConnectionImpl:sendNotificationsIfNecessary,ExtSysConnectionUpdateStatus
10.7.2.1.2	If configured to do so, the system shall issue a notification when an external connection transitions to the “Failed” state and remains there for more than a configurable connection-specific amount of time.	External Interface	Send External Connection Notification	getExternalConnectionAlertAndNotificationSettingsForm, setExternalConnectionAlertAndNotificationSettingsForm, ExternalSystemConnectionImpl:sendNotificationsIfNecessary,ExtSysConnectionUpdateStatus
10.7.2.1.3	If configured to do so, the system shall generate an External Connection Alert when an external connection transition to the “Warning” state and remains there for more than a connection-specific amount of time. Time spent in the connection failure state contributes towards the total duration in the “Warning” state.	External Interface	Create External Connection Alert	getExternalConnectionAlertAndNotificationSettingsForm, setExternalConnectionAlertAndNotificationSettingsForm, ExternalSystemConnectionImpl:sendNotificationsIfNecessary,ExtSysConnectionUpdateStatus
10.7.2.1.4	If configured to do so, the system shall issue a notification when an external connection transition to the “Warning” state and remains there for more than a connection-specific amount of time. Time spent in the connection failure state contributes towards the total duration in the “Warning” state.	External Interface	Send External Connection Notification	getExternalConnectionAlertAndNotificationSettingsForm, setExternalConnectionAlertAndNotificationSettingsForm, ExternalSystemConnectionImpl:sendNotificationsIfNecessary,ExtSysConnectionUpdateStatus
10.7.2.2	The system shall monitor and maintain the state of the connections to RITIS used for import of data from RITIS. *	EXISTING	Monitor External System Connection	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.7.2.2.1	The system shall automatically attempt to re-establish the any RITIS connection used for import of data from RITIS when it is detected to be down or when it seems unreasonably inactive *	EXISTING	Re-establish Connection To External System	
10.7.2.3	The system shall monitor and maintain the state of the connection to INRIX.	Travel Times	Import INRIX Data	
10.7.2.4	The system shall monitor and maintain the state of the connection to VECTOR.	Toll Rates	Import Vector Data	
10.7.3	The system shall allow a user to view the status of external connections established by the system.	External Interface	View External Connection Status	viewExternalConnectionStatus
10.7.3.1	The system shall display the name of the external system connection.	External Interface	View External Connection Status	viewExternalConnectionStatus
10.7.3.2	The system shall display the current status of the external system connection (OK, WARNING, or FAILED).	External Interface	View External Connection Status	viewExternalConnectionStatus
10.7.3.3	The system shall display descriptive text that provides details about the status if the status is WARNING or FAILED.	External Interface	View External Connection Status	viewExternalConnectionStatus
10.7.3.4	The system shall display the time the status transitioned into its current state.	External Interface	View External Connection Status	viewExternalConnectionStatus
10.7.3.5	The system shall display the time that status was last confirmed.	External Interface	View External Connection Status	viewExternalConnectionStatus
10.7.4	The system will allow an administrator to manage public/private key pairs for use by external client applications.	External Interface	Manage External Clients, Generate Key Pair	generateKeyPair, downloadPrivateKey
10.7.4.1	The system shall allow an administrator to add an external client application to the system.	External Interface	Add External Client	getAddEditExternalClientForm,addEditExternalClient
10.7.4.1.1	The system shall require a client application ID for the external client application.	External Interface	Add External Client	getAddEditExternalClientForm,addEditExternalClient
10.7.4.1.2	The system shall require a name for the external client application.	External Interface	Add External Client	getAddEditExternalClientForm,addEditExternalClient
10.7.4.1.3	The system shall support a description for the external client application.	External Interface	Add External Client	getAddEditExternalClientForm,addEditExternalClient
10.7.4.1.4	The system shall support contact information for the point of contact for the external client application.	External Interface	Add External Client	getAddEditExternalClientForm,addEditExternalClient
10.7.4.1.5	The system shall require a public key for authenticating the external client application.	External Interface	Add External Client	getAddEditExternalClientForm,addEditExternalClient

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.7.4.1.5.1	The system shall allow an administrator to generate a public/private key pair.	External Interface	Generate Key Pair	generateKeyPair, downloadPrivateKey
10.7.4.1.5.1.1	The system shall allow an administrator to export a generated private key file for the administrator to distribute to client system personnel.	External Interface	Generate Key Pair	generateKeyPair, downloadPrivateKey
10.7.4.1.6	The system shall allow the user to specify whether the external client is a data supplier to CHART.	External Interface	Add External Client	getAddEditExternalClientForm
10.7.4.1.7	The system shall allow the user to specify whether the external client is a consumer of CHART data.	External Interface	Add External Client	getAddEditExternalClientForm
10.7.4.1.7.1	The system shall allow the user to specify the user roles that the external client consumer will be given upon successful authentication.	External Interface	Add External Client	getAddEditExternalClientForm
10.7.4.2	The system shall allow an administrator to modify the settings for an external client application registered with the system.	External Interface	Edit External Client	getAddEditExternalClientForm
10.7.4.2.1	The system shall allow an administrator to modify the settings that were specified when adding an external client application according to requirement 10.7.4.1 and its subrequirements.	External Interface	Edit External Client	getAddEditExternalClientForm
10.7.4.3	The system shall allow an administrator to remove an external client application from the system.	External Interface	Remove External Client	removeExternalClient
10.7.4.3.1	The system shall prompt the user for confirmation before removing an external client application from the system.	External Interface	Remove External Client	removeExternalClient
10.7.4.4	The system will allow an administrator to view the external client applications that have been configured for access to CHART.	External Interface	View External Clients	viewExternalClientList,createDynList,getDynListSubjects
10.7.4.3	The system shall allow an administrator to register a public key and associated client application ID with one or more CHART services.	External Interface	Add External Client	getAddEditExternalClientForm,addEditExternalClient
10.7.4.4	The system shall allow an administrator to export a generated private key file for the administrator to distribute to client system personnel.	External Interface	Generate Key Pair	downloadPrivateKey
10.7.5	The system shall allow an administrator to maintain a list of external agencies used when importing objects from external systems.	External Interface	Configure External Agencies To Orgs Mapping	getExternalOrgToAgencyMappings,setExternalOrgToAgencyMappings

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.7.5.1	Each External Agency will be specified for use with a specific External System and should be unique with that External System.	External Interface	Configure External Agencies To Orgs Mapping	getExternalOrgToAgencyMappings,setExternalOrgToAgencyMappings
10.7.5.2	The system shall require an administrator to specify a Chart Organization to be associated with an External Agency when creating / updating the External Agency.	External Interface	Configure External Agencies To Orgs Mapping	getExternalOrgToAgencyMappings,setExternalOrgToAgencyMappings
10.8	The system shall integrate event data from external systems together with internally created CHART events.	EXISTING	N/A	
10.8.1	Upon initial receipt of event data from an external system, the system shall open a CHART event and designate that event an external event.	EXISTING	Create CHART External Traffic Event	handleExternalImport
10.8.1.1	When a candidate external event satisfies any external event import rule which specifies that an alert be sent, the system shall send an External Event Alert to the specified Operations Center referencing the new CHART copy of the external event.	External Interface	Create External Event Alert	handleExternalImport
10.8.1.2	When a candidate external event satisfies any external event import rule which indicates that the event should be marked as interesting, the system shall mark the new CHART copy of the external event as “interesting”.	External Interface	Flag Interesting RITIS Events	handleExternalImport
10.8.1.3	When a new CHART external event is created the system shall set its owning organization based on a mapping, defined in the system, between External System / agency and CHART organizations.	External Interface	Map External Agency To Org	handleExternalImport
10.8.1.3.1	When a new CHART external event is created and there is no owning organization mapped for the external system / agency, the system shall set its owning organization to a configurable default owning Organization.	External Interface	Map External Agency To Org	handleExternalImport
10.8.1.4	When a new CHART external event is created, the system shall import data into CHART which is available from the external sources.	External Interface	Create CHART External Traffic Event	Use Case only
10.8.1.4.1	When a new CHART external event is created, the system shall import the event creation time if it is available from the external source.	External Interface	Create CHART External Traffic Event	Use Case only
10.8.1.4.2	When a new CHART external event is created, the system shall import the event closed time if it is available from the external source.	External Interface	Create CHART External Traffic Event	Use Case only

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.8.1.4.3	When a new CHART external event is created, the system shall import the scene cleared time if it is available from the external source.	External Interface	Create CHART External Traffic Event	Use Case only
10.8.1.4.4	When a new CHART external event is created, the system shall import the event state if it is available from the external source.	External Interface	Create CHART External Traffic Event	Use Case only
10.8.1.4.5	When a new CHART external event is created, the system shall import the external system name.	External Interface	Create CHART External Traffic Event	Use Case only
10.8.1.4.6	When a new CHART external event is created, the system shall import the external event ID.	External Interface	Create CHART External Traffic Event	Use Case only
10.8.1.4.7	When a new CHART external event is created, the system shall import the external agency.	External Interface	Create CHART External Traffic Event	Use Case only
10.8.1.4.8	When a new CHART external event is created, the system shall import the location description if it is available from the external source.	External Interface	Create CHART External Traffic Event	Use Case only
10.8.1.4.9	When a new CHART external event is created, the system shall import the CHART event type.	External Interface	Create CHART External Traffic Event	Use Case only
10.8.1.4.10	When a new CHART external event is created, the system shall import lane status if it is available from the external source.	External Interface	Create CHART External Traffic Event	Use Case only
10.8.1.4.11	When a new CHART external event is created, the system shall import vehicles involved data if it is available from the external source.	External Interface	Create CHART External Traffic Event	Use Case only
10.8.1.4.12	When a new CHART external event is created, the system shall import geolocation data (lat/long) if it is available from the external source.	External Interface	Create CHART External Traffic Event	Use Case only
10.8.1.5	When a candidate external event satisfies any external event import rule which specifies that a notification should be sent, the system shall send a notification to the specified notification group.	External Interface	Send External Event Notification	handleExternalImport
10.8.2	Upon receipt of updates to a CHART external event data from an external system, the system shall update the corresponding CHART external event. *	EXISTING	Update Existing CHART External Traffic Event	handleExternalImport

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.8.2.1	Upon receipt of updates to a CHART external event from an external system, the system shall update the corresponding CHART external event's creation time if it is available from the external source.	External Interface	Update Existing CHART External Traffic Event	Use Case Only
10.8.2.2	Upon receipt of updates to a CHART external event from an external system, the system shall update the corresponding CHART external event's closed time if it is available from the external source.	External Interface	Update Existing CHART External Traffic Event	Use Case Only
10.8.2.3	Upon receipt of updates to a CHART external event from an external system, the system shall update the corresponding CHART external event's scene cleared time if it is available from the external source.	External Interface	Update Existing CHART External Traffic Event	Use Case Only
10.8.2.4	Upon receipt of updates to a CHART external event from an external system, the system shall update the corresponding CHART external event's traffic event state.	External Interface	Update Existing CHART External Traffic Event	Use Case Only
10.8.2.5	Upon receipt of updates to a CHART external event from an external system, the system shall update the corresponding CHART external event's location description if it is available from the external source.	External Interface	Update Existing CHART External Traffic Event	Use Case Only
10.8.2.6	Upon receipt of updates to a CHART external event from an external system, the system shall update the corresponding CHART external event's event description if it is available from the external source.	External Interface	Update Existing CHART External Traffic Event	Use Case Only
10.8.2.7	Upon receipt of updates to a CHART external event from an external system, the system shall update the corresponding CHART external event's CHART event type information.	External Interface	Update Existing CHART External Traffic Event	Use Case Only
10.8.2.8	Upon receipt of updates to a CHART external event from an external system, the system shall update the corresponding CHART external event's lane status if it is available from the external source.	External Interface	Update Existing CHART External Traffic Event	Use Case Only
10.8.2.9	Upon receipt of updates to a CHART external event from an external system, the system shall update the corresponding CHART external event's vehicles involved if it is available from the external source.	External Interface	Update Existing CHART External Traffic Event	Use Case Only

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.8.2.10	Upon receipt of updates to a CHART external event from an external system, the system shall update the corresponding CHART external event's geolocation data (lat/lon) if it is available from the external source.	External Interface	Update Existing CHART External Traffic Event	Use Case Only
10.8.3	Upon receipt of an event closure notification from an external system, the system shall close the corresponding CHART external event.	EXISTING	N/A	
10.8.4	The system shall archive external event data into the CHART archive database after the archive timeout expires following closure of the external event.	EXISTING	N/A	
10.8.5	The system shall allow a suitably privileged user to close external events. (This is expected to be done only by very few highly privileged users, and only for events which are "stale" and believed to be truly closed by the originating agency.) *	EXISTING	N/A	
10.8.6	The system shall allow a suitably privileged user to request a list of external traffic events.	EXISTING	N/A	
10.8.7	The system shall optionally hide external events associated with an external connection that has been detected to be down for an administrator configurable period of time.	EXISTING	N/A	
10.8.8	The system shall allow a suitably privileged user to manage zero or more rules for determining which external events the system is to import into CHART (and by implication, which external events to exclude from CHART).	External Interface	Configure External Event Import Rules, View Event Import Rules	viewTrafficEventInclusionRules
10.8.8.1	The system shall allow a suitably privileged user to create and store zero or more external event import rules.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.1	The system shall allow a suitably privileged user to indicate if an External Event Alert is associated with the rule and, if so, which operations center to send it to when the rule is satisfied.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.2	The system shall allow a suitably privileged user to indicate if the satisfaction of a rule should cause the external event to be marked as "interesting."	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3	The system shall allow a suitably privileged user to define criteria all of which must be satisfied for the external event import rule to be satisfied.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.8.8.1.3.1	The system shall support the use of geographical coordinates as an external event import rule criterion.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.1.1	The system shall support the use of one or more named geographical areas (see Section 1.1.1.3.2) as a criterion in an external event import rule. The intent is that an external event is said to satisfy an external event import rule's geographic coordinate's criterion if its latitude and longitude are found within any of the rule's named geographical areas.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.1.1.1	The system shall support an "Empty" geographical area which matches an external event's lack of latitude and/or longitude values	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.1.1.2	The system shall support an "Any" geographical coordinate criterion which matches any external event's geographical coordinates.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.2	The system shall support the use of one or more U.S. states as an external event import rule criterion.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.2.1.1	The system shall support an "Empty" U.S. state criterion which matches an external event's lack of a state value.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.2.2	The system shall support an "Any" state which matches any external event's state.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.3	The system shall support the use of route type, as defined in CHART, as an external event import rule criterion.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.3.1	The system shall support an "Empty" route type criterion which matches an external event's lack of a route type value.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.2.2	The system shall support an "Any" state which matches any external event's state.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.4	The system shall support the use of partial matching of text in key text fields of an external event as an external event rule criterion.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.4.1	The system shall support the use of partial matching text in the external event's name as an external event rule criterion.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.8.8.1.3.4.2	The system shall support the use of partial matching text in the external event's description as an external event rule criterion.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.4.3	The system shall support the use of partial matching text in the external event's route number as an external event rule criterion.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.4.4	The system shall support the use of partial matching text in the external event's county as an external event rule criterion.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.5	The system shall support the use of lanes closed as an external event rule criterion.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.5.1	The system shall support an "Empty" lanes closed which matches an external event's lack of a lanes closed value.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.5.2	The system shall support an "Any" lanes closed value which matches any external event's closed lanes value.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.3.6	The system shall support the use of one or more CHART Event Types as an external event rule criterion.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.1.4	The system shall allow a suitably privileged user to indicate if a notification is associated with the rule and, if so, which notification group to notify when the rule is satisfied.	External Interface	Add Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.8.8.2	The system shall allow a suitably privileged user to delete an external event import rule.	External Interface	Delete Event Import Rule	removeTrafficEventInclusionRule
10.8.8.3	The system shall allow a suitably privileged user to modify an external event import rule by providing the same criteria options as when creating a rule.	External Interface	Edit Event Import Rule	getAddEditTrafficEventInclusionRuleForm,addEditTrafficEventInclusionRule
10.9	The system shall integrate DMS data from external systems together with internally created CHART DMS data.	External Interface	Import RITIS DMS Data	Use Case Only
10.9.1	Upon receipt of an update to a candidate external DMS, the system shall update the corresponding external DMS, if it exists.	External Interface	Update Existing CHART External DMS	handleDMSImportTask ExternalDMS:updateStatus DMSControlClassDiagram2
10.9.2	The system shall allow a suitably privileged user to manage the importing of a DMS from an external system into the CHART system.	External Interface	Configure Candidate External DMSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates, getCandidates

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.9.2.1	The system shall allow a suitably privileged user to view a list of candidate external DMSs.	External Interface	Configure Candidate External DMSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates, getCandidates
10.9.2.1.1	The system shall display the name of each candidate external DMS	External Interface	Configure Candidate External DMSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm
10.9.2.1.2	The system shall display the agency of each candidate external DMS.	External Interface	Configure Candidate External DMSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm
10.9.2.1.3	The system shall display the location description of each candidate external DMS.	External Interface	Configure Candidate External DMSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm
10.9.2.1.4	The system shall indicate if each candidate external DMS is already an external DMS in the CHART system.	External Interface	Configure Candidate External DMSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.9.2.1.5	The system shall indicate if each candidate external DMS was previously rejected for importing.	External Interface	Configure Candidate External DMSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.9.2.1.6	The system shall allow the user to filter the list of candidate external DMSs as an aid in deciding which to import	External Interface	Filter Candidate External Device List	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.9.2.1.6.1	The system shall allow the user to filter the list of candidate external DMSs by zero or more Agencies. Filtering by zero Agencies means to not filter by Agency.	External Interface	Filter Candidate External Device List	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.9.2.1.6.2	The system shall allow the user to filter the list of candidate external DMSs by zero or more named geographical area (see Section 1.1.1.3.2). Filtering by zero named geographical areas means not to filter by geographical area.	External Interface	Filter Candidate External Device List	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.9.2.1.6.3	The system shall allow the user to filter the list of candidate external DMSs by whether user-entered text partially matches the device name, device description, location description, county, or route name.	External Interface	Filter Candidate External Device List	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.9.2.1.6.4	The system shall allow the user to filter the list of candidate external DMSs by whether they are included in CHART's list of external DMSs.	External Interface	Filter Candidate External Device List	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.9.2.1.6.5	The system shall allow the user to filter the list of candidate external DMSs by whether they are excluded from CHART's list of external DMSs.	External Interface	Filter Candidate External Device List	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.9.2.1.6.6	The system shall allow the user to filter the list of candidate external DMSs by whether they are neither included nor excluded from CHART's list of external DMSs (i.e. an import decision has not yet been made).	External Interface	Filter Candidate External Device List	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.9.2.2	The system shall allow a suitably privileged user to create an external DMS from a candidate external DMS.	External Interface	Configure Candidate External DMSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates DMSControlCD2
10.9.2.2.1	When a new external DMS is created the system shall set its owning organization based on a mapping, defined in the system, between External System / agency and CHART organizations.	External Interface	Map External Agency To CHART Org	setCandidates
10.9.2.2.1.1	The system shall use a configurable default owning organization for the DMS if a mapping does not exist.	External Interface	Map External Agency To CHART Org	setCandidates
10.9.2.3	The system shall allow a suitably privileged user to delete an external DMS from the CHART system (Note: This does not prevent its use as a candidate external DMS)..	External Interface	Delete External DMS	Use case only
10.9.2.3.1	When a user deletes an external DMS, the system shall mark the corresponding candidate external DMS as "excluded."	External Interface	Delete External DMS	Use case only
10.9.2.4	The system shall allow a suitably privileged user to indicate if an external DMS was rejected for import.	External Interface	Configure Candidate External DMSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.1	The system shall integrate TSS data from external systems together with internally created CHART TSS data.	External Interface	Import RITIS Detector Data	Use Case Only
10.10.1	Upon receipt of an update to a candidate external TSS, the system shall update the corresponding external TSS, if it exists.	External Interface	Update Existing CHART External TSS	handleTSSImportTask
10.10.2	The system shall allow a suitably privileged user to manage the importing of a TSS from an external system into the CHART system.	External Interface	Configure Candidate External TSSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.10.2.1	The system shall allow a suitably privileged user to view a list of candidate external TSSs	External Interface	Configure Candidate External TSSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm
10.10.2.1.1	The system shall display the name of each candidate external TSS.	External Interface	Configure Candidate External TSSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.10.2.1.2	The system shall display the agency of each candidate external TSS.	External Interface	Configure Candidate External TSSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm
10.10.2.1.3	The system shall display the location description of each candidate external TSS.	External Interface	Configure Candidate External TSSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm
10.10.2.1.4	The system shall indicate if each candidate external TSS is already an external TSS in the CHART system.	External Interface	Configure Candidate External TSSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm
10.10.2.1.5	The system shall indicate if each candidate external TSS was previously excluded for importing.	External Interface	Configure Candidate External TSSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm
10.10.2.1.6	The system shall allow the user to filter the list of candidate external TSSs as an aid in deciding which to import.	External Interface	Configure Candidate External TSSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm
10.10.2.1.6.1	The system shall allow the user to filter the list of candidate external TSSs by zero or more Agencies. Filtering by zero Agencies means to not filter by Agency.	External Interface	Filter Candidate External Device List	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.10.2.1.6.2	The system shall allow the user to filter the list of candidate external TSSs by zero or more named geographical areas (see Section 1.1.1.3.2). Filtering by zero named geographical areas means to not filter by geographical area.	External Interface	Filter Candidate External Device List	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.10.2.1.6.3	The system shall allow the user to filter the list of candidate external TSSs by whether user-entered text partially matches the device name, device description, location description, county, or route name.	External Interface	Filter Candidate External Device List	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.10.2.1.6.4	The system shall allow the user to filter the list of candidate external TSSs by whether they are included in CHART's list of external TSSs.	External Interface	Filter Candidate External Device List	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.10.2.1.6.5	The system shall allow the user to filter the list of candidate external TSSs by whether they are excluded from CHART's list of external TSSs.	External Interface	Filter Candidate External Device List	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.10.2.1.6.6	The system shall allow the user to filter the list of candidate external TSSs by whether they are neither included nor excluded from CHART's list of external TSSs (i.e. an import decision has not yet been made)	External Interface	Filter Candidate External Device List	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.10.2.2	The system shall allow a suitably privileged user to create an external TSS from a candidate external TSS.	External Interface	Configure Candidate External TSSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.10.2.2.1	When a new external TSS is created the system shall set its owning organization based on a mapping, defined in the system, between External System / agency and CHART organizations.	External Interface	Map External Agency To CHART Org	setCandidates
10.10.2.2.1.1	The system shall use a configurable default owning organization for the TSS if a mapping does not exist.	External Interface	Map External Agency To CHART Org	setCandidates
10.10.2.3	The system shall allow a suitably privileged user to delete an external TSS from the CHART system. (Note: This does not prevent its use as candidate external TSS.)	External Interface	Delete External TSS	Use Case only
10.10.2.3.1	When a user deletes an external TSS, the system shall mark the corresponding candidate external TSS as “excluded.”	External Interface	Delete External TSS	Use Case Only
10.10.2.4	The system shall allow a suitably privileged user to indicate if an external TSS was rejected for import.	External Interface	Configure Candidate External TSSs	getExternalDeviceQueryForm,submitExternalDeviceQueryForm,ExternalDeviceManagerClasses, searchCandidates, setCandidates
10.11	The system shall integrate with INRIX for the purpose of requesting travel times for routes configured within CHART.	Travel Times	Import INRIX Data	
10.11.1	The system shall pull data from the INRIX system web service at a configurable periodic interval.	Travel Times	Import INRIX Data	
10.11.1.1	The system shall raise an External Connection Alert if the data cannot be successfully retrieved from the INRIX system web service for a configurable period of time.	Travel Times	Import INRIX Data, Create External Connection Alert	
10.11.1.2	The system shall raise an External Connection Alert if the data retrieved from the INRIX system web service does not conform to the documented format.	Travel Times	Import INRIX Data, Create External Connection Alert	
10.11.1.3	The system shall raise an External Connection Alert if the data retrieved from the INRIX system web service does not contain one or more links that have been associated with CHART travel routes.	Travel Times	Import INRIX Data, Create External Connection Alert	
10.11.2	The system shall archive data used to create travel time messages.	Travel Times	Archive Link Data,, Archived Route Data	
10.11.2.1	The system shall archive all raw data received from the INRIX system web service for a configurable period of time.	Travel Times	Archive Link Data	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.11.2.2	The system shall archive, for a configurable period of time, all data for each configured travel route sufficient to determine the travel time messages that were displayed at a particular time for that travel route.	Travel Times	Archive Route Data	
10.11.3	The system shall allow an administrator to import link definitions from the INRIX distribution CD via an offline process.	Travel Times	Import INRIX Links	
10.11.3.1	Each import of INRIX link definitions from the distribution CD shall fully replace all previously imported INRIX link definitions. (Note: INRIX guarantees that roadway links will never be deleted (even links that are supplanted by a collection of smaller links that lie within the older, larger link). So this should not invalidate any existing CHART Travel Routes as long as the bounding rectangle includes all roadway links currently configured in those existing CHART travel routes (see 10.11.3.2).)	Travel Times	Import INRIX Links	
10.11.3.2	The system shall import only those INRIX link definition whose start and end points lie within a configurable bounding rectangle.	Travel Times	Import INRIX Links	
10.12	The system shall integrate with VECTOR for the purpose of asynchronously receiving toll rate information for routes configured within CHART.	Toll Rates	Import Vector Data	
10.12.1	The system shall allow the VECTOR system to post toll rate update documents to a web service hosted at a configurable publicly accessible IP address and port.	Toll Rates	Import Vector Data	
10.12.1.1	The system shall verify that posted data has originated from the VECTOR system by requiring that the posted data be digitally signed with a previously provided private key.	Toll Rates	Authenticate External System	
10.12.1.2	The system shall validate all data posted by the VECTOR system against the published XSD.	Toll Rates	Import Vector Data	
10.12.2	The system shall return a response XML document each time data is received from the VECTOR system.	Toll Rates	Import Vector Data	
10.12.2.1	The system shall return a success code and a list of accepted toll routes each time the VECTOR system successfully posts a toll rate update.	Toll Rates	Import Vector Data	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.12.2.1.1	The system shall set the External Connection state to OK when a valid, complete message is received from the VECTOR system.	Toll Rates	Import Vector Data	
10.12.2.2	The system shall return a failure code and a list of error code/error message pairs each time the VECTOR system posts invalid data.	Toll Rates	Import Vector Data	
10.12.2.2.1	The system shall reject any toll rate data that does not have a digital signature with an authorizationError error code with corresponding error text.	Toll Rates	Import Vector Data	
10.12.2.2.2	The system shall reject any toll rate data that contains a digital signature that cannot be read with the previously provided public key by returning an authorizationError error code with corresponding error text.	Toll Rates	Import Vector Data	
10.12.2.2.3	The system shall reject any toll rate data that does not validate correctly against the published XSD by returning an invalidXML error code with corresponding error text.	Toll Rates	Import Vector Data	
10.12.2.2.4	The system shall reject any toll rate update that has a startDateTime that is more than a configurable number of minutes in the future by returning an invalidStartDateTime error code with corresponding error text.	Toll Rates	Import Vector Data	
10.12.2.2.5	The system shall reject any toll rate update that has an expirationDateTime that is not later than the posted startDateTime by returning an invalidExpirationDateTime error code with corresponding error text.	Toll Rates	Import Vector Data	
10.12.2.3	The system shall set the External Connection state to WARNING each time the VECTOR system posts data that fails validation or authorization.	Toll Rates	Import Vector Data	
10.12.2.4	The system shall set the External Connection state to WARNING each time the VECTOR system posts data does not contain data for any toll route (start id/destination id pair) that is associated with a CHART system travel route.	Toll Rates	Import Vector Data	
10.12.2.5	The system shall set the External Connection state to FAILED if it does not receive any data from the VECTOR system for a configurable period of time (expected to be on the order of 1 hour).	Toll Rates	Import Vector Data	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.12.2.6	If configured to do so, the system shall generate an alert if it does not receive any data from the VECTOR system for a configurable period of time (expected to be on the order of 1 hour).	Toll Rates	Import Vector Data	
10.12.2.7	If configured to do so, the system shall issue a notification if it does not receive any data from the VECTOR system for a configurable period of time (expected to be on the order of 1 hour).	Toll Rates	Import Vector Data	
10.12.2.8	The system shall reject in full any toll rate update from the VECTOR system that contains any errors and shall not use any toll rate from said update document. (Other than sending the appropriate error code in the XML response, the system shall otherwise react as if no document had been sent at all.)	Toll Rates	Import Vector Data	
10.12.3	The system shall completely replace all current toll rates with the posted data each time the VECTOR system successfully posts a toll rate update.	Toll Rates	Provide Toll Data to CHART Travel Routes	
10.12.3.1	The system shall immediately clear the current toll rate for any VECTOR system toll route that is not included in the most recently posted VECTOR system toll rate update document.	Toll Rates	Provide Toll Data to CHART Travel Routes	
10.12.3.2	The system shall immediately clear the current toll rate for any VECTOR system toll route that is included in the most recently posted VECTOR system toll rate update document but does not include the optional rate element.	Toll Rates	Provide Toll Data to CHART Travel Routes	
10.12.3.3	The system shall immediately set the current toll rate for every VECTOR system toll route that is included in the most recently posted VECTOR system toll rate update which contains a valid value in the optional rate element.	Toll Rates	Provide Toll Data to CHART Travel Routes	
10.12.4	The system shall support an optional field that specifies the expiration date/time of all toll rates in each posted VECTOR system update document.	Toll Rates	Expire Toll Rates	
10.12.4.1	The system shall discontinue use of any toll rate posted by the VECTOR system whose expiration date/time has expired.	Toll Rates	Expire Toll Rates	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.12.4.2	If configured to do so, the system shall generate an alert each time a toll rate that is in use expires.	Toll Rates	Create Toll Rate Alert	
10.12.4.3	If configured to do so, the system shall issue a notification each time a toll rate that is in use expires.	Toll Rates	Send Toll Rate Notification	
10.12.4.4	The system shall not automatically expire toll rates posted by the VECTOR system without an expiration date/time.	Toll Rates	Expire Toll Rates	
10.12.5	The system shall maintain an online cache of all posted toll rate updates for each route for a configurable period of time. (Expected to be on the order of one hour.)	Toll Rates	View Travel Route Details	
10.12.6	The system shall archive all toll rate updates posted by the VECTOR system and shall keep them available for offline inquiry for a configurable period of time. (Expected to be on the order of one week.)	Toll Rates	Archive Toll Rate Data	
10.12.7	The system shall archive, for a configurable period of time, all data for each configured travel route sufficient to determine the toll rate messages that were displayed at a particular time for that travel route.	Toll Rates	Archive CHART Travel Route Toll Rate Data	
10.12.8	The system shall allow the VECTOR system to post toll rate update documents to a backup web service hosted at a configurable publicly accessible IP address and port.	Toll Rates	Import Vector Data	
10.12.8.1	The system shall support the generation of alerts and notifications when the backup web service is posted to.	Toll Rates	Import Vector Data	
10.12.8.1.1	If configured to do so, the system shall generate an External Connection Alert each time the VECTOR system posts toll rate data to the backup web service.	Toll Rates	Import Vector Data	
10.12.8.1.2	The system shall allow an administrator to configure minimum interval that must expire between issued notifications to inform that the VECTOR system is posting toll rate data to the backup web service.	Toll Rates	Import Vector Data	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.12.8.1.3	If configured to do so, and if the minimum interval has expired since the last such push notification, the system shall issue a notification when the VECTOR system posts toll rate data to the backup web service.	Toll Rates	Import Vector Data	
10.13	CHART Export Service. The system shall provide a service for the purpose of providing CHART information to authorized third-party clients.	External Interface	Provide Data to External Systems	Use case only
10.13.1	The external export service shall export data for CHART devices and Traffic Events.	External Interface	Provide Data to External Systems	Use Case only
10.13.1.1	The external export service shall export data on CHART DMSs to authorized third-party clients who request it.	External Interface	Provide DMS Data to External Systems	
10.13.1.2	The external export service shall export data on CHART TSSs to authorized third-party clients who request it.	External Interface	Provide Detector Data To External Systems	
10.13.1.3	The external export service shall export data on CHART HARs to authorized third-party clients who request it.	External Interface	Provide HAR Data To External Systems	
10.13.1.4	The external export service shall export data on CHART SHAZAMs to authorized third-party clients who request it.	External Interface	Provide SHAZAM Data To External Systems	
10.13.1.5	The external export service shall export data on CHART Cameras to authorized third-party clients who request it. [FUTURE, NEW]	FUTURE	N/A	
10.13.1.6	The external export service shall export data on CHART Traffic Events to authorized third-party clients who request it.	External Interface	Provide Traffic Event Data To External Systems	
10.13.2	The system shall allow third-party clients to post request documents to a web service hosted at a configurable publicly accessible IP address and port.	External Interface	Provide Data To External Systems	
10.13.2.1	The system shall verify that posted data has originated from an authorized third-party system by requiring that the posted data be digitally signed with a previously provided private key and include a client application ID authorized to be used with that key.	External Interface	Authenticate External System	
10.13.2.2	Each request document shall contain a CHART user login.	External Interface	Authenticate External System	

Req No.	Requirement	Type	Use Case(s)	Additional Element(s)
10.13.2.3	Response data shall be limited to include only data that the provided CHART user login has sufficient functional rights to access.	External Interface	Provide Data To External Systems	
10.13.2.4	The system shall validate all data requests posted by a third-party system against the published XSD.	External Interface	Provide Data To External Systems	
10.13.3	The system shall return a response XML document each time a data request is received from a third-party system.	External Interface	Provide Data To External Systems	
10.13.3.1	The system shall return an error code and corresponding error text for each data request that contains errors.	External Interface	Provide Data To External Systems	
10.13.3.2	The system shall return valid XML as defined by the system ICD and XSD for each valid data request.	External Interface	Provide Data To External Systems	

7 Acronyms/Glossary

Arbitration Queue	A prioritized queue containing messages for display or broadcast on a traveler information device.
CCTV	Closed Circuit Television
CHART	Coordinated Highways Action Response Team
CORBA	Common Object Request Broker Architecture. CORBA is the CHART application's architecture for distributed computing.
CORBA Event	A CORBA mechanism using which different CHART components exchange information without explicitly knowing about each other.
DMS	Dynamic Message Sign
EORS	Emergency Operations Reporting System
FMS	Field Management Server
GUI	Graphical User Interface
HAR	Highway Advisory Radio
HIS	Highway Information Systems
IOR	Interoperable Object Reference
ISDN	Integrate Services Digital Network
ISS	Information System Specialists
JRE	Java Run-time Environment
MDOT	Maryland Department of Transportation
MDSHA	Maryland State Highway Administration
MdTA	Maryland Transportation Authority
NTCIP	National Transportation Communications for ITS Protocol
POTS	Plain Old Telephone Service
RITIS	Regional Integrated Transportation Information System
RTMS	Remote Traffic Microwave Sensor
SHA	State Highway Administration
SOC	Statewide Operations Center
Synchronized HAR	A HAR entity that is comprised of one or more HAR transmitters (also known as constituent HARs).
TSS	Traffic Sensor System
TTS	Text to Speech